January/February 1981

Single Copy Price $7.50

I/O News Volume One, Number Three

FORTRAN SUBROUTINES

INTRODUCTION TO C

CROMEMCO SOFTWARE

The OFFICIAL PUBLICATION OF THE INTERNATIONAL ASSOCIATION OF CROMEMCO USERS

Cromemco Again Leads The Industry With New C Compiler

When Bell Laboratories unveiled the C programming language in 1975, it quickly became one of the most popular languages in academic computing circles. More recently its popularity has grown in industrial and commercial applications as well. C has been implemented on minicomputers, like the DEC PDP-11, on mainframes, including the IBM-370 and Honeywell 6070, and now Cromemco offers C on its complete line of microcomputers.

What makes C so very useful and well-liked by programmers is its simplicity. It is also a very rich and expressive language conducive to structured programming style. In fact, with C it is very often not necessary to use an intermediate English-like description of a solution to a programming problem, a common first step in the development of a program; one simply forms a solution expressed in C. C contains the types of modern flow-control statements which are needed to construct readable and well-structured programs: **for** (loop on condition with an automatic end-of-loop statement), **while** (loop with test at top), **do...while** (loop with test at bottom), **if...else...**, **switch** on expression value (known as the **case** statement in many languages), **break**, **continue**, and **return**.

C offers a variety of data types: **char**, **int**, **unsigned**, **long**, **float**, **double**, pointers to any type of object, and aggregates (arrays, structures, and unions) of the basic types of data or of other aggregates. The C

Continued on page 22

Using String Arrays in Basic

By Chris Rook
Technical Sales Manager, Cromemco, Inc.

The implementation of the string function in Cromemco BASIC has a distinct advantage over the "string arrays" implemented in some other versions of BASIC. In particular, Cromemco strings can be of arbitrary size (limited only by available memory size) while string arrays are limited in size (usually to 128 bytes). When programming in Cromemco BASIC, it is important to understand the distinction between a Cromemco string and the older type of string array.
For example:

Dim A$(99)

In Cromemco BASIC this would allocate 100 bytes of space for the single dimension string "A$". In a string array this would allocate 990 bytes of space for the two dimen-

Continued on page 22

STRESS

A Program for Linear Static Analysis of Engineering Structures

Authors: Dubravko Nardini, Ph.D.
Nikolaj Ivancic, M.Sc.
Miljenko Srikoc, B.Sc

I. Introduction

STRESS is the acronym for **STR**uctural **E**ngineering **S**ystems **S**olver. It is a computer program which performs linear static analysis of engineering structures. In spite of the appearance of a number of structural analysis packages, STRESS remains one of the most popular programs in that field among the practicing structural engineers.

The development of STRESS

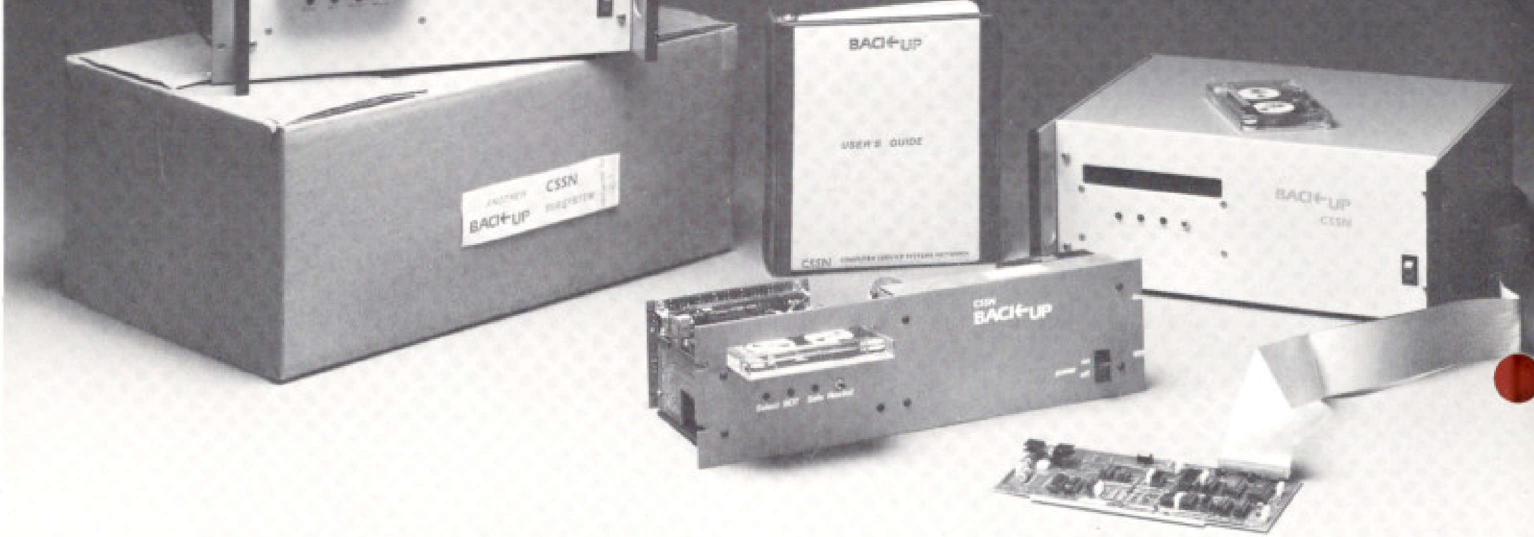
Continued on page 14

BACK←UP

Don't let a hard disk be the soft spot in your data security

The availability of fast, reliable, high capacity hard disk storage for the S-100 computer market has created a wave of excitement. It has also underscored the somber necessity for a reliable means of backup. No serious application is practical without a dependable, economical method for backup and archival of critical on-line data.

Now, CSSN breaks the barriers to hard disk applications by offering the complete hardware and software solution for data security...



Operates with all CROMEMCO systems using either CDOS tm or CROMIX tm

Software provides:

- File by file save and restore operation using standard CDOS tm file naming conventions
- Tape files that are totally O.S. independent
- Files grouped in logical savesets
- Selective restore can key off of save date
- Includes command file and log file facilities
- All Lifeboat Software now available on CSSN Backup cartridges

Hardware provides:

- 6400 BPI cartridge tape drive with power supply
- Up to 13.4 megabytes per tape
- S/100 interface card
- Rack mount unit or table top enclosure

See your Cromemco dealer, or write:

CSSN COMPUTER SERVICE SYSTEMS NETWORK

120 BOYLSTON STREET • FOURTH FLOOR • BOSTON MASSACHUSETTS 02116 • (617) 482-2340 • TWX (710) 321-1913 INCORPORATED

table of contents

January/February 1981 Volume I, Number Three

I/O News The Official Publication of The International Association of Cromemco Users is available through membership in the association. Editorial and advertising policies are designed for the enlightenment of the members in regard to new uses for, and developments of, Cromemco products and other products compatible with Cromemco systems.

COVER FEATURES

ARTICLES & FEATURES

I/O News (ISSN 0274-9998) is published bi-monthly by The International Association of Cromemco Users (a California corporation), P.O. Box 17658, Irvine, CA 92713. General offices are at 4750 Von Karman Avenue, Suite 500, Newport Beach, CA 92660. Telephone: (714) 955-0432. Controlled Circulation Postage Paid at Santa Ana, CA. POSTMASTER: Send address changes to I/O News, P.O. Box 17658, Irvine, CA 92713.

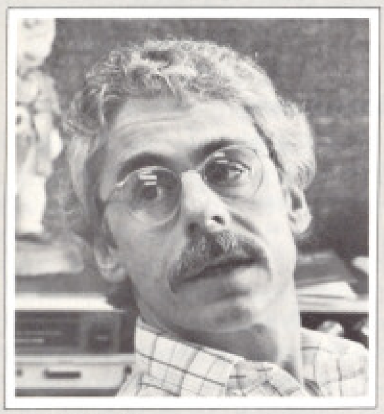
Subscriptions to I/O News are entered with membership in The IACU. Yearly memberships may be purchased for $35 (U.S. delivery address), $41 (delivery address in Canada or Mexico), and $48 (other international delivery address). Contact IACU for multi-year membership rates. Back issues of I/O News are available for $7.50 per issue. Please note: all prices are in U.S. dollars.

Return postage must be included with all manuscripts and photos submitted if they are to be returned. The IACU and I/O News accept no responsibility for the return of unsolicited materials.

Richard Kaye
Editor and Publisher

Lynn Platzek
Editorial Assistant

output...



RICHARD KAYE

LYNN PLATZEK

GUEST COLUMNIST

Last issue we introduced the concept of a monthly "Guest Columnist" to offer ideas as to what we may expect from micros in the future. Since then, Cromemco has made a very generous offer. If there is enough interest by members in futurist activities of this sort, Cromemco will put up some prizes — some very substantial prizes — and we can turn this into a contest.

The rules are simple. Each issue, for the next six issues, we will feature the best ideas submitted. The ideas should revolve around how micros will affect our lives in the next 20 years, and should be as specific as possible. At the end of the contest period, we will have an independent panel of judges determine the winners based on such criteria as feasibility, ease of use, general acceptance of the idea by the public, simplicity of manufacture, practicality, scope and social effects of the idea, and so on. To give you an idea as to how big the "computer revolution" is, a business acquaintance told me that he heard a speech by a vice president of Texas Instruments who stated: "By 1990 there will be 100,000 bytes of memory for every man, woman, and child in the entire world." Food for thought, eh?

From chatting with members from all over the world, I believe there are a lot of worthy ideas out there. This is the place to showcase those ideas, and perhaps win some great prizes. How about it?

ARTICLES IN THE MILL

Your response has been quite encouraging and we are now starting to build an inventory of articles for future issues. They include such diverse topics as Data Encryption, Speech Processing, Educational Applications, TWX Interfacing, Micro-Monitoring of Hospital Patients, Computer Sculpture, A Program for Depreciation Schedules, Financial Tracking of a Multi-Currency, Multi-Bank Corporation, and even a software package, for banks that produce Federal Income Tax Data for Individuals on Both Monthly and Year-End Statements.

We have many more articles in the works, and are still contacting members who indicated they would be willing to submit material for publication. Some of the most useful articles are those dealing with practical applications in a given engineering, scientific, or business environment. We would find your practical experiences very valuable and solicit your editorial contributions.

GOOD NEWS

We have entered, full swing, into the new year and are discovering some exciting software developments. Through your calls and letters, we have found that most of you are looking for a certain package to perform specific tasks. And, we have been able to find the right source in many cases. But, the really "good news" is that a great deal of software that has been under development for two or three years is starting to emerge.

There are new accounting packages, inventory packages, job costing packages, medical and dental packages, banking packages, and financial planning packages. There are also new versions of software, modified to run under CDOS and especially CROMIX.

With the emergence of this software, we enter a new phase in the evolution of micros. Much of this generation of software offerings will be advertised on the pages of I/O News. We encourage you to try these sources first. Also, if you find software that you can recommend, please let us know. We will contact the supplier and attempt to arrange evaluations so that others can know about it, too.

Richard Kaye
Editor & Publisher

Lynn Platzek
Editorial Assistant

Introduction to C

by David K. Ellis

Since C is a relatively new language, many people are not yet familiar with it. This article is designed to provide an introduction to the C language for Cromemco users who may be considering using it on their systems.

A C program consists of one or more separately-compiled modules which are combined to form a single executable file. Each module of a C program consists of definitions of symbols and macros, external data definitions, and 0 or more functions. External data can be **extern** (visible to all modules in the program), or **static** (visible only to functions within that one module). Functions are not nested, as are procedures in block-structured languages such as Algol and Pascal. Each function is visible from all functions in all modules in the program. A function definition consists of declarations for up to 32 arguments followed by the function body, which is a single compound statement containing all of the local data definitions and statements needed to carry out the task of the function. Local data are data which are visible only to the function enclosing them, and are classified as either **auto** or **static. auto** data exist only during the time in which the function is executing; they cease to exist and their values are lost when the function returns control to the function which called it. **static** data exist for the lifetime of the entire program and their values are never lost. Subsequent calls to a function will find the final **static** values which existed during the function's previous incarnation.

All data can be given initial values at the time they are declared. The initialization of **static** and **extern** data occurs once, conceptually at the start of execution of the program. **auto** data is initialized to the declared values each time the enclosing function is executed.

Functions may be called recursively, either directly or indirectly, and all functions may return a single value, although this is not required by C. Arguments are passed to C functions by value, not by reference. This means that the value of an argument, not its address, is passed to a function. The function therefore cannot inadvertently change the value of an argument, a common source of programming errors in languages such as Fortran which do pass arguments by reference. A multitude of programmers have discovered the hard way that constants in such languages are not really constant after a function has forgotten itself and assigned a new value to an argument which started life as a constant.

Sometimes, however, it is necessary for a function to change a variable outside of its scope or to return more than a single value, and C allows this. When an array name is used as an argument to a function, C converts this to a pointer to the first element of the array. The function can then declare the formal argument to be an array and load new values into the array, which in reality is the array in the calling function. Also, variables called pointers can be passed to a function which can then assign new values to the objects pointed to by the pointers.

Data

Cromemco C has the following basic data types:

char
short
int
unsigned
long
float
double.

A **char** contains a single byte, which can be an ASCII character or a number in the range 0-255. A **char** is never negative. **short** is the same as **int,** which can contain a signed value in the range −32768 to 32767. An **unsigned** variable holds an unsigned integer in the range 0 to 65,535. A **long** variable is used when a larger integer value is required; the approximate range is −2E9 to 2E9. **float** and **double** variables hold BCD (Binary Coded Decimal) floating point values in the range ± 9.99E−65 to ± 9.99E+62. The digit precision for **float** is 6, and 14 for **double.**

C Statements

Statements in C are terminated, not separated, by semicolons. C will continue to scan a statement until it finds a semicolon, so a missing semicolon usually causes a syntax error. Statements can be grouped together by braces, { }, into a single block, or compound statement. A compound statement can be used wherever a single statement can be used. Whether or not braces are better than BEGIN-END is frequently debated; they are certainly less bother to type and read. (We won't even suggest where the braces should be placed or with what indentation—each programmer is convinced that his convention makes the most sense.)

C offers the following flow-control statements. The list is short, easily learned, but not lacking in utility.

;

The semicolon can be used by itself as a place-taking null statement.

break;

The **break** statement causes control to pass from the enclosing **do, for, switch,** or **while** statement.

continue;

The **continue** statement causes control to pass to the loop continuation part of the enclosing **do, for,** or **while** statement.

```
do
        stmt
while (expr);
```

The **do** statement repeatedly executes **stmt** while **expr** is true. The test is done at the end of the loop.

```
for (expr1; expr2; expr3)
        stmt
```

The **for** statement evaluates **expr1** then repeatedly executes **stmt** and evaluates **expr3** so long as **expr2** is true. The test is done before each loop.

goto label; Yes, C does have a **goto.** It is rarely used.

```
if (expr)
        stmt1
else
        stmt2
```

If **expr** is true, **stmt1** is executed, otherwise **stmt2** is executed. The **else** stmt2 part is optional.

```
return;
or
return expression;
```

The **return** statement returns control to a calling function. It can optionally return a value of the type declared for the enclosing function.

```
switch (int-expr) {
        case const-expr: stmt
        ...
        default: stmt
 }
```

The **switch** statement permits a multi-way branch. **int-expr** is evaluated and control proceeds to the **stmt** preceded by a **const-expr** equal to **int-expr.** If no match exists, control proceeds to the **default** statement. If there's no **default**, control falls through.

```
while (expr)
        stmt
```

The **while** statement repeatedly executes **stmt** so long as **expr** is true. The test is done before each loop.

A symbol followed by a colon, **:**, can precede a statement for use as the destination of a **goto.**

C Operators

One of the strengths of C is its large number of operators. There are operators for performing assignment, arithmetic, logical, relational, and unary operations. It has the usual **+**, **−**, **/**, and *****, as well as **%**, remainder division. C also has:

bitwise operators:
& and,
^ exclusive or,
| inclusive or;
logical operators:
&& and,
|| or;
shift operators:
<< left shift,
>> right shift;
relational operators:
< less than,
> greater than,
<= less than or equal to,
>= greater than or equal to,
== equal to,
!= not equal to;

C contains several unary operators, some of which are unique to C:

* indirection,
& address of an object,
− arithmetic negation,
! logical negation,
~ one's complement,
(type) cast,
sizeof size in bytes.

A cast converts the expression which follows it to the specified type.

Two extremely useful unary operators are **++** (increment) and **−−** (decrement). When **++** precedes an operand, the operand is incremented by one before its value is used in an expression. When **++** follows an operand, the value is used before the operand is incremented. This permits the incrementing of indices to be combined with other expressions to yield concise, yet readable, code, as shown by the following lines, which obtain a character from an array and increment the subscript to be ready for the next access:

```
c = array[i++]; ...
```

C contains the usual assignment operator =, along with compound assignment operators formed by one of the operators, + − * / % >> << & | followed by =. An example will illustrate the effect of a compound assignment operator:

```
array[sub] += 3;
```

This expression adds three to the value of array[sub], where array might be an array of integers. At first glance this form of assignment might look odd, but one quickly realizes how much easier this is to use than the Fortran

```
array(sub) = array (sub) + 3
```

Another aspect of assignment operators in C is that they are treated as any other binary operator. This means that an assignment is an expression which can be nested inside a more complicated expression, yielding efficiency both to the C source program and the generated object code. This extremely useful treatment of assignment is found in only a few other programming languages.

An example which shows how to use this feature is

```
while ( (c = getchar() ) != EOF )
        process ( c );
```

This example shows a loop which will get one character from the file **stdin** (described later in this article) and process it unless **getchar** returns EOF (a defined value) instead of a character. A less expressive language would need to have two reads, one preceding the loop to handle the case in which EOF is detected on the first read, and another inside the loop to get a character for the next test. The parentheses around the assignment are necessary because assignment has a lower operator precedence than !=.

Pointers, Arrays, Structures, and Unions

C contains variables called pointers which are defined to contain the actual memory address of other objects. A pointer can be loaded with the address of a data object or a function, the value of another pointer, or an integer. Expressions can then obtain or change the value of the object "pointed to" by the pointer by use of the indirection operator, *. This allows one function to pass a pointer to a mass of data, such as an array, to another function, rather than passing the entire mass of data to

Continued on next page

Introduction to C

Continued from page 7

the function. C in fact has no facility for passing the values of a block of data around between functions. The second function can then perform its task with the given data and return. A pointer can be incremented, and when it is, C conveniently scales the increment by the size of the object to which the pointer is declared to point. For instance, an integer requires two bytes of memory, so when a pointer to an integer is incremented by 1, as it might be when using a pointer to step through an array of integers, C adds a value of 2 to the pointer, thereby releasing the programmer from the responsibility of knowing how large the various data types are.

Arrays in C can contain any of the basic types of data, or constructed types such as other arrays, structures, or unions. For example,

```
char aname[10][20]
```

is a 10 x 20 array of characters. Array elements are numbered starting with 0.

A structure is a collection of data objects, such as variables, pointers, arrays, other structures, or unions, under one name. C structures are similar in function to Pascal and Cobol records, although they cannot be moved around or compared with a single statement as can Cobol records.

A simple structure might be declared:

```
struct {
        char flag;
        int count;
} sname;
```

sname is the name of the structure, which can be used with the dot operator, **.**, to reference the members of the structure: **sname.flag** references the member of **sname** whose name is **flag.** One can also declare a pointer to a structure, then access the members using the pointer, which is really a necessary feature since a structure itself cannot be passed as an argument to a function. Suppose that **ps** is a pointer to **sname**; then **ps->flag** references the member **flag.**

A union permits the same block of storage to be accessed as one type of data at one time and as a different type of data at another, depending on the programmer's intention. A union is essentially a structure all of whose members have the same offset. A simple union might be declared:

```
union {
        char flag;
        struct {
                int i;
                long 1;
        } data;
} unumber;
```

One might conceivably store an integer value into **unumber.data.i**, then set **unumber.flag** to 'i' to indicate that the union contains an integer.

Input and Output

There are no I/O statements in C. All input and output is done by calling functions which exist in the C library.

There are 30 I/O functions in the C library, and 10 other functions which perform useful tasks. In addition, the C library contains 46 CROMIX system call interface functions for those users who wish to use CROMIX system calls from their C programs.

There are I/O functions which create, open, and close files, and functions which read or write single bytes, lines, or block of bytes to files. A program can position to any location within a file for the next read or write. The library contains a very useful and frequently-used function, **printf**, which performs formatted output, converting the values of variables from internal form to human-readable form using a control string containing conversion specifications. A much-appreciated feature of this function is that the control string and the values to be output are separate arguments to **printf**. The control string is written unchanged except for the conversion specifications, which are preceded by the percent symbol, **%**. A conversion specification calls for the conversion and insertion of one of the values from the remaining arguments. For example,

```
printf( "the values are %d, %f, and %02x.", i, j, k);
```

where i = 25, J = 10.4, and k = 09 hexadecimal, will write

```
the values are 25, 10.4, and 09.
```

There is also a function, **scanf**, which performs formatted input, converting human-readable values to the appropriate values in C variables. **printf** and **scanf** have analogs which perform formatted I/O on files (**fprintf** and **fscanf**) and on strings (**sprintf** and **sscanf**).

stdin, stdout, and stderr

stdin, **stdout**, and **stderr** are three files which are always open and available to a C program. **stdin** is an input file which supplies characters from a standard input device, usually the user's terminal keyboard. **stdout** is an output file which delivers characters to the standard output device, again usually the user's terminal. **stderr** is a file traditionally reserved for the receipt of error messages, which are usually delivered to the user's terminal. Data from or destined for these files can be redirected using the CROMIX I/O redirection capability. The use of < **file** in the execution command line of a C program which reads from **stdin** means that the program will get data from **file** whenever it attempts to read **stdin**. Similarly, the use of > **file** in the execution command line of a C program which writes characters to **stdout** means that the program will write the data onto **file** whenever it attempts to write to **stdout**. **file** is automatically created with zero length before the first I/O access when output is redirected. Many existing C programs perform all of their I/O using **stdin** and **stdout**, which are redirected each time the user executes the program.

Currently characters written to **stderr** always appear on the user's terminal when running under the CROMIX system and in the same file with **stdout** when running under CDOS. CDOS itself cannot redirect I/O. A C program which is linked to execute under CDOS performs its own redirection when called on to do so.

A Sample C Program

A listing of **clist.c** follows this article. This program is supplied as part of the C package, and is used to list one or more source files to the terminal with line numbers and page breaks.

clist contains two functions: **main**, which does all the work, and **testabort**, which is called to check for user abort before each line is written. **clist** calls several functions which are found in the C library: **fopen**, **fclose**, **fgets**, which reads a line, **fputs**, which writes a line, **putchar**, which writes a single character, **getmode**, which uses the CROMIX **.getmode** system call to get the status of the user's terminal, and **printf**, which is used here to write the line number followed by a tab character (\t).

The variables **argc** and **argv** allow a C program to use the arguments (in this case file names) from the command line which invokes execution.

The variable **fp** is a pointer to a special structure called a FILE structure which is used to indicate to the I/O functions which file to read or write.

C treats an assignment operator as any other binary operator, which means that an assignment is an expression which can itself be used or tested. We see an example of this in **clist**, line number 44:

```
(abort = testabort() ) = = 0
```

The variable **abort** is assigned a value from the call to the function **testabort**, and that value is then tested as part of the conditional expression.

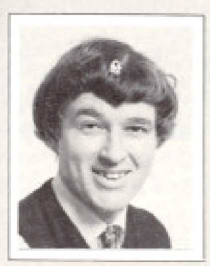
The symbols /* and */ are used to delimit comments, which may occupy several lines. Indentation has been used to indicate to the human eye which statements are subordinate to others, although C ignores white space (tabs and spaces) in most cases.

We see that C contains a means to define symbols like CR so that C programs need not contain "magic" numbers which have meaning only to the author of the program.

Summary

C is a very useful general-purpose programming language whose popularity is spreading rapidly. It is concise and expressive. The language offers many features which make programs easy to write, read, and maintain.

Continued on next page



About the Author

Dave Ellis joined Cromemco in 1978 to manage the Cobol product, and his interest and responsibilities rapidly broadened to include the Fortran and Ratfor packages, diagnostics software, and now the C product. Dave is currently enrolled in the Computer Science Master's program at the University of Santa Clara.

```
/* clist

   Copyright (c) 1980, by Cromemco, Inc., All Rights Reserved

   This program lists one or more C source files, with line
   numbers and page breaks at end of page and end of each
   file.  clist aborts if the user presses Control-C.

*/

#control nsource
#include "stdio.h"
#include "modeequ.h"
#control source

#define CR       13      /* carriage return */
#define EXPCHAR   8      /* expanded-print character for dot-matrix */
#define FF       12      /* form feed */
#define LF       10      /* line feed */
#define SP       32      /* space */


main( argc, argv )

unsigned argc;
char      *argv[];

{
   static unsigned linenum,
                   linecnt,
                   fnum = 0;
   static char     lnbuf[ 258 ];
   static int      numchars,
                   abort = 0,
                   i;
   static FILE     *fp;


   while ( ++fnum < argc && abort == 0 ) {
        fp = fopen( argv[fnum], "R" );
        if  ( fp ) {
            linenum = 0;
            linecnt = 56;
            while( fgets( lnbuf, 256, fp ) && (abort = testabort()) == 0 ) {
                if ( ++linecnt > 56 )  {
                    putchar( FF );
                    putchar( EXPCHAR );
                    putchar( SP );
                    fputs( argv[fnum], STDOUT );
                    putchar( CR );
                    putchar( LF );
                    putchar( LF );
                    linecnt = 1;
                }
                printf( "%+ 5d\t", ++linenum );
                fputs( lnbuf, STDOUT );

            }  /* end while_fgets */

            fclose( fp );

        }  /* end if_fp */

   }  /* end while_fnum */

   putchar( FF );

}  /* end main */

testabort()       /* returns non-zero if cntl-c was hit, 0 if not */
{
        return ( getmode( STDIN, md_status ) & st_abort );
}
```

Fortran Subroutines For Opening Files (That Try Harder)

Jerome J. Tieman

Have you ever been annoyed at yourself when you tried to run one of your FORTRAN programs, and it bombed with an EF (end of file) error because you had your data file on the wrong disk drive? Well if so — or if you want to know how to open a file without necessarily knowing or caring which drive it's on — the following will be of interest to you.

The problem described above would be trivial to solve if FORTRAN had the ability to ask the file system whether or not a particular file existed before attempting to open it, but the unfortunate facts of life are that there is no such command in the language. It is, however, possible to accomplish the objective by exploiting the very same EF error that keeps reminding us of our human failings. Even though FORTRAN cannot read the directories of your disks, the read statement does have an END = STATEMENT# option which can be used to detect which disk a particular file is on. This option is typically used to divert the program flow when the last record in a file has been read, so as to break out of a read loop. In what follows, we will describe a different use, and thereby show how to open and read a file without knowing which disk it's on. The syntax of a FORTRAN READ statement that uses this option is:

```
READ(LUN,FORMAT#,END=STMT#)(LIST OF ITEMS)
```

Where LUN is the logical unit number of the file, and the other two parameters of the read statement are statement numbers referring to the FORMAT statement for the record to be read and the optional statement number for handling the end of file condition.

When you try to open a file in FORTRAN on the wrong disk, the program cheerfully goes ahead and opens a file of that name on the disk you told it to (expecting that you obviously want to write something on it), but until something has been written, it has zero length. When you try to read from it, you get the aforementioned end of file error, and back you go to square one! Unless, of course you have used the END = STATEMENT# option.

The first of the two subroutines given below will try first to open the file you are looking for on the logged in disk, and then, if it is not found there, systematically go looking for it first on the A drive and then B,C,D, etc. until you run out of drives or until it finds the file. This is accomplished by using the variable I as the drive specifier in the CALL OPEN statement instead of a constant, and assigning 0,1,2,3 etc. to I before opening the file. That is, we use:

```
CALL OPEN(LUN,IFCB,I)
```

instead of

```
CALL OPEN(LUN,IFCB,0)
```

Here, LUN is the logical unit number by which the file is to be referenced in all subsequent FORTRAN statements, IFCB is an array that contains an expanded version of the filename, and the third parameter is the disk drive selector.

In SUBROUTINE OPENF(LUN,ILINE,IERR) given below, a particular disk selection is made and, after opening the file, an attempt is made to read a record from it. If the file is empty, an end of file condition occurs on the very first record, and the program flow for this case causes I to be incremented, the file that was just opened (on the wrong disk) to be closed, and a new open attempted with the next value of I. When I gets larger than the largest valid disk drive number in your particular system, you give up and go back to get a more reasonable filename. The calling program finds this out by testing the value of IERR, which is set to zero when all is well, to 3 when the file is not found and to 4 when the string reformat subroutine didn't like the filename you gave it. (It found one or more unprintable control characters.) Hopefully, the file you are looking for exists somewhere, and when you finally get there, it will not return the end of file condition on the very first record and the program flow will proceed usefully onward.

When you check your directories after using this method, you will find entries for "the filename that wasn't there" on all the disks that were checked unsuccessfully. These entries don't do much harm; since they all have zero length, they don't use any disk space, and the next time you run your program, you will get exactly the same behavior. That's because we are not just looking for any old file with the desired name, we are specifically looking for one that has something written on it. There is, however, one caveat that should be mentioned! Namely, there is a bug in many of the FORTRAN compiler versions which bites you when you try to open a file and there is no directory space. So make sure you have room for several more directory entries on **all** your disks before using them with a program that incorporates this technique.

As indicated in the comments in the program listing, you have two options concerning the first line in the file: If you want to pass the opened file in the pristine state to the calling program, you can REWIND it before returning. In this case, the calling program will re-read the first line, and the net effect will be the same as a simple open

Continued on next page

Fortran Subroutines

Continued from page 11

statement. But there is another alternative which has some advantages, and that is to return after reading the first line without rewinding the file. Why do this? There are two reasons: First, it is a very good idea to include a title line in your files anyway, and it is more efficient to skip past it in the file opening subroutine than to repeat the needed code everywhere a file is opened. Title lines are good because they help you keep track of what your files contain and verify that the correct files were used even if all else fails. (Have you ever tried to invent an eight letter filename that was supposed to remind you that the contents came from a run where 4 two digit parameters had specific values and that the date of the run was June 14 1980? [And it didn't!] You can incorporate all that and even a hint as to what you had in mind in the first place when you have a whole line to play with.) The second reason you may want to skip the first line when reading data has to do with a conflict of opinion about what constitutes an end of record symbol. In the Industry Standard file system, an end of record is a carriage return-line feed pair, while in FORTRAN, it is just a carriage return. Thus, FORTRAN will return a linefeed as the first character of every line EXCEPT THE FIRST. This will obviously wreak havoc with any FORMAT statements that try to read data from both the first line and any following lines. The simple way out is to not include data on the first line, but to use this line as a title line instead. If you agree that this is a good idea, then let your file opener subroutine read the title, print it out for you and skip over it as it does in the example below.

As another example of how a disk searching file opener can be used to advantage, consider the possibility that you may want to run a program that reads several groups of parameters and data, and that you want to assign specific choices to some (but not all) of these. Any data or parameter values that are not explicitly given are to be assigned standard default values. This can be accomplished by assigning each group of parameters or data values a descriptive filename, and putting the desired default values in files having these names on the disk drive with the highest number. Then, if you put the explicitly given data in files of the same names on lower numbered drives, these will be chosen wherever they exist, while the default data will get chosen for any of the files that are not found on the lower numbered drives. This problem often arises in computer modeling of complex systems, and the method described above is a convenient way to generate runs where several families of parameters are held constant from run to run while other families are changed.

The second of the two subroutines presented below is named REFORM(ILINE,IFCB,IERR). It is useful even if you don't want or need a disk searcher. It simply accepts a filename and extent in the industry standard form and expands it to the form required by FORTRAN. It also filters out any control characters (which make erasing a file nearly impossible) and protects you from getting lower case filenames as well. These are also hard to erase, and are a problem because if you make an error when you type in a filename, directory entries for that exact name will be created.

The calling sequence involves two arrays named ILINE and IFCB as well as our friend IERR mentioned above. The first array is for input, and it contains an ASCII string denoting the filename in the industry standard form, while the second is an output array which returns a FORTRAN compatible filename string. The input array can be defined either by compiling it in as follows:

```
      REAL ILINE(3)
      DATA ILINE /'FILE','NAME','.EXT'/
```

Or the filename can be gotten from the operator console using

```
      LOGICAL ILINE(72)
      READ(3,10) ILINE
 10   FORMAT(72A1)
```

Note that the length of the input array need not correspond to the length of the payload it contains, and that it can be longer or shorter than 12 characters as long as it can hold the filename plus four characters for the period and the extent.

```
C     THIS SUBROUTINE TAKES A LUN AND A CPM FORMAT
C     FILENAME, CONVERTS IT TO THE FORTRAN FORM AND
C     TRIES TO OPEN THE FILE ON EITHER THE LOGGED DISK,
C     OR DISK A OR DISK B, ETC.
C     THIS ROUTINE TRIES TO READ THE FIRST LINE IN THE FILE
C     TO SEE IF IT IS THERE, SO MAKE THE FIRST LINE A TITLE
C         OR A DUMMY.
C     IF FILE IS NOT ON THE DEFAULT DISK(S), AN EMPTY
C     DIRECTORY ENTRY GETS CREATED ON IT. IF
C     DIRECTORY IS FULL — BAD NEWS!!!
      SUBROUTINE OPENF(LUN,ILINE,IERR)
      LOGICAL ILINE(72),IFC(11)
C     ILINE IS A BYTE (LOGICAL*1) ARRAY. CAN BE READ AS
C     FORMAT (##A1) WHERE ## IS ANY VALUE LARGE
C     ENOUGH TO HOLD THE FULL NAME.
          FULL NAME
C     FIRST REFORMAT THE FILENAME
      CALL REFORM(ILINE,IFCB,IERR)
C     CHECK FOR BAD FILENAME
      IF(IERR.GT.2)RETURN
C     NOW OPEN THE FILE
C     TRY THE LOGGED DISK FIRST
      I=0
      GOTO 45
   50 I=I+1
C     WASN'T THERE SO QUIT AND TRY THE NEXT DISK
      ENDFILE LUN
C     SOONER OR LATER, YOU SHOULD GIVE UP
      IF(I.GE.5)GOTO100
   45 CALL OPEN(LUN,IFCB,I)
C     READ THE DUMMY FIRST LINE
      READ(LUN,22,END=50)ILINE
   22 FORMAT(72A1)
C     AND PRINT IT OUT
      WRITE(3,23)ILINE
   23 FORMAT(1X,72A1)
C     INDICATE SUCCESS TO CALLING PROGRAM AND RETURN
C     IF DUMMY LINES AREN'T A GOOD IDEA, THEN REWIND LUN
C     HERE SO THE CALLING PROGRAM GETS THE FIRST LINE
C     FILE (BUT DON'T FORGET THE LINE-FEED PROBLEM)
```

```
C         REWIND LUN
          IERR=0
          RETURN
C         NO MORE DISKS TO TRY
    100   WRITE(3,24)
     24   FORMAT('  FILE DOES NOT EXIST')
          IERR=3
          RETURN
          END
C         THIS SUBROUTINE REFORMATS THE FILENAME AND
C         EXTENT.
C         ILINE IS THE INPUT STRING AND IFCB IS THE OUTPUT
C         STRING
          SUBROUTINE REFORM(ILINE,IFCB,IERR)
          LOGICAL ILINE(12),IFCB(11)
C         SCAN THE FILENAME FIELD
          DO 10    I = 1,8
C         CHECK FOR PERIOD DELIMITER
          IF(ILINE(I).EQ.46)GOTO 20
C         NOT A PERIOD SO CHECK THE CHARACTER AND MOVE IT
C         FIRST CHECK FOR UNPRINTABLES
          IF(ILINE(I).LT.32)GOTO15
C         CHANGE LOWER CASE TO UPPER
          IFCB(I)=ILINE(I)
          IF(ILINE(I).GT.95)IFCB(I)=ILINE(I).AND.95
     10   CONTINUE
C         IF WE GET HERE, THERE WERE EIGHT LETTERS IN THE
C         NAME
C         NEXT CHARACTER IS EITHER A DOT OR THE EXTENT
          I=9
          IF(ILINE(I).EQ.46)GOTO 20
          J=I
          GOTO 30
     15   WRITE(3,21)
     21   FORMAT('   UNWORTHY FILENAME')
          IERR=4
          RETURN
C         AT STATEMENT 20, I POINTS TO THE DOT
     20   J=I+1
C         J NOW POINTS TO THE EXTENT
     25   IF(I.EQ.9)GOTO 30
C         SHORT FILENAME SO PAD WITH BLANKS
          IFCB(I)=32
          I=I+1
          GOTO 25
C         NOW MOVE THE EXTENT
     30   DO40 K=9,11
          IF(ILINE(J).LT.32)GOTO 15
          IFCB(K)=ILINE(J)
          IF(ILINE(J).GT.95)IFCB(K)=ILINE(J).AND.95
          J=J+1
     40   CONTINUE
          IERR=0
          RETURN
          END
```

BIOGRAPHICAL SKETCH of Dr. Jerome J. Tiemann (Version 80.11)

The threads that led to this article lead back at least to the early 1950's when Jerry came to Stanford to get a PhD in Physics. That was where he first learned to program a Computer, and it was no mean feat. There were no high level languages at all in those days, and one had to deal with rather bizarre instruction sets with one's bare hands, so to speak. The Computers available at the Stanford Computation Center were the IBM 605 and, later on, one of the first IBM 650's. This three address programmer's nightmare combined the limitations of a 2000 word magnetic drum memory with the inconvenience of an I/O system that required one to wire up one's own plug board. But an even more significant thread was the fact that Jerry was more often seen in the company of the EE grad students than with his fellow Physicists. His continuing interest in EE (which was one of his two Minor fields) led to a close and continuing relationship with the Stanford EE faculty. It was this relationship that led to the meeting of Jerry and a young grad student named Roger Melen in the early 1970's. Their mutual interests at the time were centered around Integrated Circuits for Analog Signal Processing, but when Roger evolved into Computers, it only increased their area of mutual interest.

Jerry was one of Cromemco's earliest customers, both for his own personal computer as well for those used "at work." Work, for Jerry is still pretty much the same as it was back at Stanford. He is on the Scientific Staff of the General Electric Corp. R/D Ctr., and he works on electronic systems, solid state electronic circuits, and novel electronic devices for implementing them. He holds over 50 patents, is the author of over 80 technical articles, and is a Fellow of both the IEEE and the American Physical Society.

STRESS

Continued from page 1

began in 1962, under the direction of Prof. S.J. FENVES, project staff including Prof. R.D. LOGCHER, Prof. S.P. MAUCH, K.F. REINSCHMIDT and R.L. WANG. The whole project took place at M.I.T., under general supervision of Prof. J.M. BIGGS, and the biggest supporters were the FORD Foundation and the Department of Defense. The first implementation was written for the IBM 7094 computer. Since that time many various implementations have been made for different computers ranging from a small IBM 1130, up to the large UNIVAC 1100 series. Every such implementation has some "local properties," or additions to the "standard STRESS." The purpose of this article is to describe the AMS (AgroMarketing Software) implementation, which is (at least to our knowledge) one of the first micro computer versions of STRESS.

II. STRESS — brief description

Treating STRESS as a black box, one can say that it is a processor acting on some input data, and producing the required output. The input consists of a sequence of statements, in free format, describing the particular engineering structure, with terminology which is entirely in the engineer's language. The output is composed of various tables containing forces and deformations within the structure.

More precisely, the term structure is used instead of "framed" structure. That means that the structure is composed of members (elements) that can be represented by their centroidal axis, and analysed as line elements. (In simpler words, we are dealing with structures which may be represented as a composition of so called beam and truss elements.) Such a structure may be subjected to static external forces, prescribed displacements and (or) temperature changes. Those forces can be applied to so called joints (that are interconnections between elements), or distributed along elements. The solution to the described problem or linear elasticity is obtained by the matrix stiffness (displacement) method. Being able to solve small, medium and large structural problems with equal efficiency, is an additional advantage of STRESS.

III. STRESS input language

As already mentioned, input to STRESS consists of a sequence of statements using common engineering terminology. The words used in STRESS will be printed in capitals & in this article. This description of language is brief, without explaining all details:

1. Header statement:

STRUCTURE text

text (characters) that follow the keyword STRUCTURE on the same line is used as a name of the structure

2. Size descriptors:

NUMBER OF JOINTS i
NUMBER OF SUPPORTS i
NUMBER OF MEMBERS i
NUMBER OF LOADINGS i

integer numbers following each of above descriptors are used to define the actual size of the problem.

3. Process descriptors:

TYPE type
TABULATE category
SELECTIVE OUTPUT
PRINT category

These statements describe the internal procedures used for particular problems. The following can be supplied instead of: type — one type — one of five possible types: plane truss, space truss, plane frame, space frame, plane grid. category — any of the following keywords: data, forces, reactions, displacements

4. Structural data descriptors:

JOINT COORDINATES
JOINT RELEASES
MEMBER INCIDENCES
MEMBER PROPERTIES
MEMBER RELEASES
CONSTANTS

These statements are used to describe the structure providing information about its geometry, topology, load-deflection relationships, and local and global releases.

5. Loading data descriptors:

LOADING
JOINT LOADS
JOINT DISPLACEMENTS
MEMBER LOADS
MEMBER DISTORTIONS
LOADING COMBINATION
COMBINE

These statements describe the loading applied to the structure, in terms of external forces and possibly prescribed displacements.

6. Termination descriptors:

SOLVE
SOLVE THIS PART

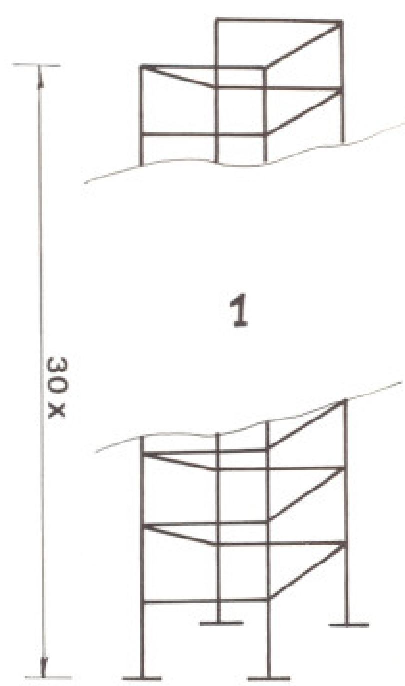
These statements are used to terminate the input phase of STRESS (and normally to start the solution phase).

To stay within normal limits for an article of that type, we shall not give the precise description of those statements. Instead, a brief example is presented, which will show their usage, and more, the syntax of STRESS language.

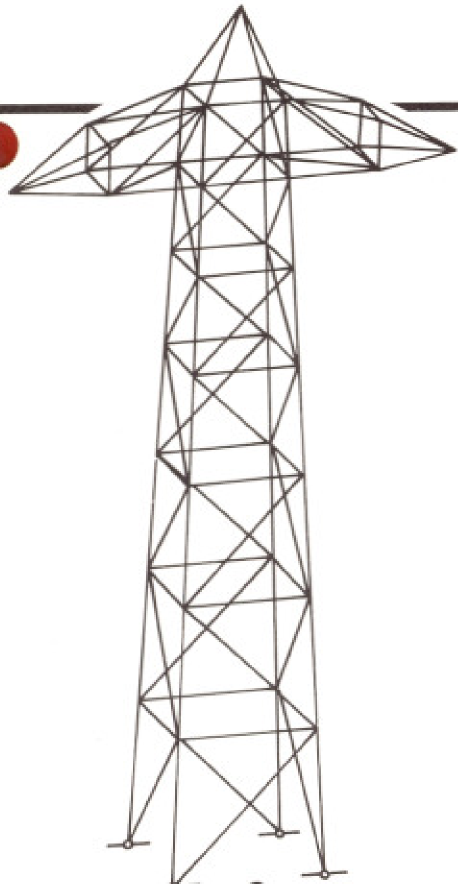
IV. Brief example

It must be pointed out that the presented material is not complete in any sense — the character of this article is purely informative. We still have the feeling that some example, how to use STRESS, could be helpful. Therefore, we have chosen the following small example:

The structure is a plane frame subjected to horizontal (wind) forces and dead and live load (snow) — see figures.



Schematics of structure with applied forces



2

Topological schematic of structure

The input data looks like this (lines starting with * are treated as comments and ignored by STRESS processor) :

```
STRUCTURE Two bay three storey
plane frame
#
* Size descriptors
#
NUMBER OF JOINTS 11
NUMBER OF SUPPORTS 3
NUMBER OF MEMBERS 13
NUMBER OF LOADINGS 3
*
* Process descriptors
#
TYPE PLANE FRAME
TABULATE ALL
*
* Structural data descriptors
#
JOINT COORDINATES
1 × 0. Y 0. S THRU 3 DX 8.0
4 × 0. Y 5. THRU 6 DX 8.0
7 × 0. Y 9. THRU 9 DX 8.0
10 × 0. Y 13. THRU 11 DX 8.0
JOINT RELEASES
3 MOMENT Z
MEMBER INCIDENCES
1 1 4 8
9 4 5 10
11 7 8 12
13 10 11
MEMBER PROPERTIES PRISMATIC
* COLUMNS - 60/60 CM
1 THRU 8 AX 0.36 IZ 0.0108
* BEAMS - 40/60 CM
9 THRU 13 AX 0.24 IZ 0.0072
CONSTANTS E 3000000. ALL
*
* Loading data descriptors
#
LOADING 1 VERTICAL LOAD
MEMBER LOADS
9 THRU 13 FORCE Y UNIFORM W
  -4.5
LOADING 2 HORIZONTAL WIND
  FORCES
JOINT LOADS
4 FORCE × 12.0
7 FORCE × 14.0
10 FORCE × 8.0
LOADING COMBINATION 3
  VERTICAL + WIND
COMBINE 1 1. 2 1.
*
* Termination descriptor
#
SOLVE
```

V. Execution phases

As already shown, STRESS has two execution phases which may be summarized as this:

1. Input analysis phase
2. Solution phase

It is the first phase, which made STRESS so popular — namely the idea of defining the input data in such a natural way that absolutely no prior computer knowledge is needed. The consequence of this is certainly the fact that STRESS runs longer than some other program performing the same job, if that program has the "standard" input (composed of numbers only, which must be entered in some rigid format).

Three types of information are allowed as input data to STRESS — integer numbers, floating point numbers and ASCII character strings — words in STRESS terminology. Separate conversion rules are applied to each of these categories, with the following general idea: every floating point number has to be stored, and the integers are used to determine where. Words may be some data to store, description of process, or data type information. The problem how to determine the meaning of words is solved very simply — by implementing the dictionary (i.e. a list of allowable words) which is used to match the word just read with its internal code.

The described method allows STRESS statements to be processed in an almost arbitrary order, without any rigidity. Besides that, the concept of internal dictionary is flexible enough to enable modifications to STRESS language without much effort.

The method of solution used in STRESS is commonly known as the "stiffness method," where the primary unknowns are displacements of the joints of the structure. The overall stiffness matrix, relating displacements to applied forces, is assembled directly from individual member matrices, employing the so called "Direct Stiffness" technique. A system of simultaneous linear algebraic equations thus obtained is solved by process of elimination, yielding the displacements of the joints. By substituting these displacements into the local force-displacements relationships, internal member forces are computed as a final result.

VI. A M S micro STRESS implementation

Due to the appearance of CROMEMCO's Overlay Linker (developed exclusively for CROMEMCO at AMS), the full implementation of STRESS became feasible. The current version of AMS micro STRESS (1.06) is divided into ten segments of code with rather sophisticated overlaying scheme. The differences between AMS and "standard" versions described above are minimal — to demonstrate it, here is the list of all acceptable statements:

STRUCTURE xxx
NUMBER OF JOINTS i
(maximal joint no. 127)
NUMBER OF SUPPORTS i

Continued on next page

NUMBER OF MEMBERS i
(maximal member no. 250)
NUMBER OF LOADINGS i
TYPE PLANE TRUSS
TYPE PLANE FRAME
TYPE PLANE GRID
TYPE SPACE TRUSS
TYPE SPACE FRAME
TABULATE
(any combination of words ALL, FORCES, REACTIONS, DISPLACEMENTS, MEMBER)
JOINT COORDINATES
(automatic data generation available)
JOINT RELEASES
MEMBER INCIDENCES
(linear generation available)
MEMBER RELEASES
MEMBER PROPERTIES PRISMATIC
FLEXIBILITY GIVEN BETA
STIFFNESS GIVEN BETA
CONSTANTS
LOADING xxx
JOINT LOADS
MEMBER LOADS
any combination of words FORCE, MOMENT, UNIFORM, CONCENTRATED, LINEAR)
JOINT DISPLACEMENTS
MEMBER DISTORTIONS
MEMBER END LOADS
LOADING COMBINATION
COMBINE
SOLVE

The following table contains some relevant parameters for serious program evaluation:

where the following annotation is used:

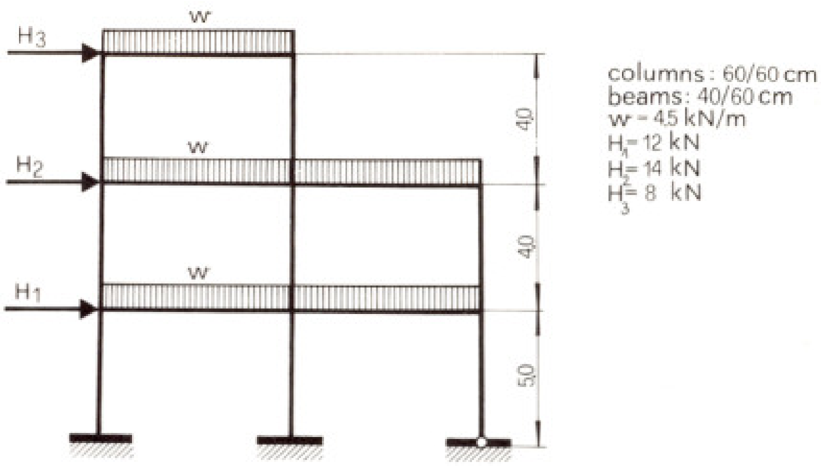
for problem description

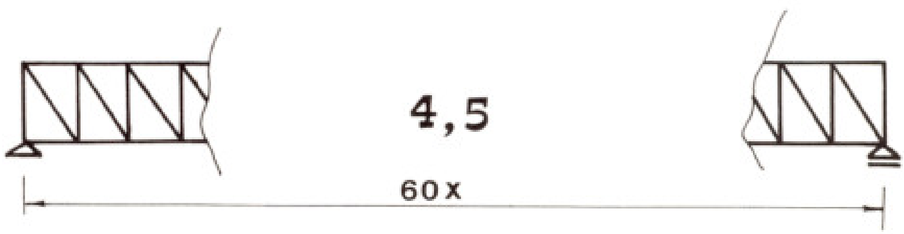
- NJ ... Number of joints
- NM ... Number of members
- NF ... Number of freedoms (equations)
- BW ... Bandwith of matrix
- NL ... Number of loadings

benchmark variables

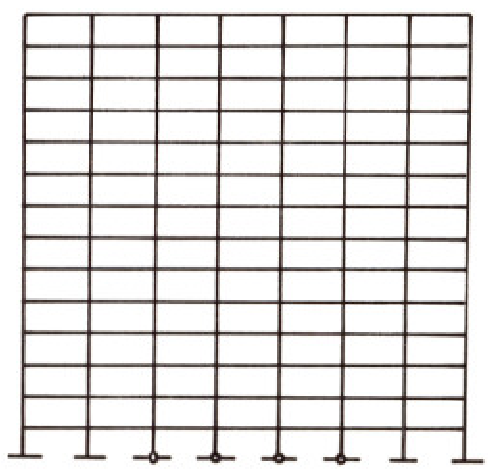
- CT ... Computer time (minutes)
- DS ... Disk space (Kilobytes)

Rough sketches of those structures are numbered according to their appearance in the above table. Structures 3 and 5 have the same geometry (once treated as truss, the other time as frame). The similar situation is with structures 4 and 6 — the first is 10 storey, 6 bay frame, the second 15 storey, 8 bay frame.

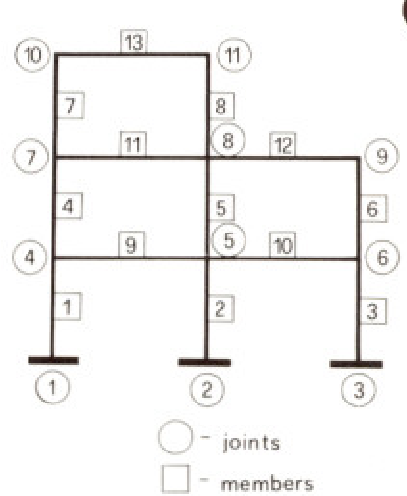


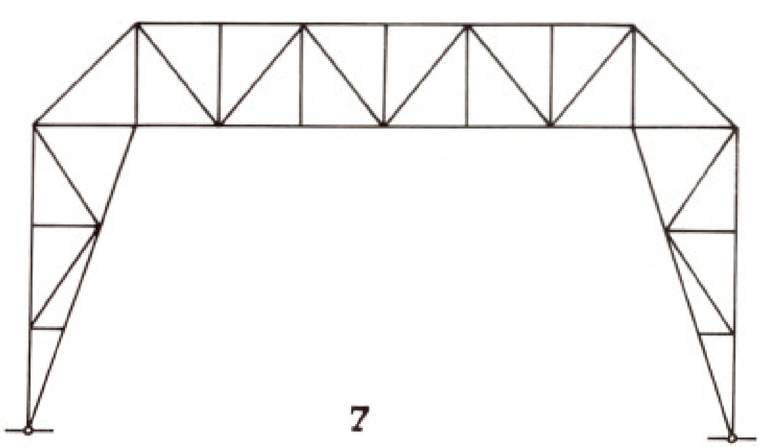


4,5



6





7

The same version of program runs under CROMIX operating system. Due to numerous I/O operations in utilizing the disk storage as a scratch file, a sophisticated buffering scheme has been incorporated into STRESS. This makes the program almost as fast under CDOS as under CROMIX (increase in speed in the latter case is only about 15 percent). Considerable improvements in program efficiency can be obtained on a system with a hard disk — at least for a factor of two. Nevertheless, the given table, with benchmark data proves that this is not the ("conditio sine qua non".)

VII. Hardware requirements

This first version of AMS micro STRESS is dedicated to users of CROMEMCO computers only. Depending on customer reactions and demand, there may appear some CP/M version too, but the present version will not work outside CS-3, or CS-2 using CDOS or CROMIX. Although not necessary, the 3102 terminal is recommended. Because of STRESS's STRESS's complexity, 64K memory is needed, but full hard disk configuration with 3355A printer is supported (i.e. 17K CDOS).

To obtain reasonably formatted output, a printer which can execute FF (Form-feed) is recommended too. The program does not engage the printer "on line", but rather saves the output on a user defined file. In that way, the printing may take place at other times and on possibly other configurations.

VIII. Future developments

Many useful features are already in process of implementation: various data generation schemes, special micro STRESS data entry program, direct coupling with new (not released yet) AMS civil engineering program for design of concrete cross-sections, and maybe the most attractive thing — the dynamic analysis of structures.

For any further information about micro STRESS, please contact (by phone, telex or letter) the following address:

AGROMARKETING
B. Adzije 7/1
41000 ZAGREB
YUGOSLAVIA
tel (041) 417662, tlx 21741 yu am zg

Num.	Structure type	NJ	NM	NF	BW	NL	CT	DS
1	Space frame	124	239	720	24	1	185.37	304
6	Plane frame	120	210	340	24	1	60.42	126
5	Plane frame	122	241	363	6	1	44.63	106
3	Plane frame	66	110	180	18	3	28.62	118
4	Plane truss	122	241	241	4	1	14.55	58
2	Space truss	43	120	117	12	1	8.90	48
7	Plane truss	26	49	48	6	2	4.33	24

About the Authors

Dubravko Nardini is the Associate Professor at the Structural Mechanics Department, University of Zagreb, Yugoslavia. The main field of his research is dynamics of structures, and some of his most recent works are published internationally. Dr. Nardini has a very broad experience with STRESS, which has started some 9 years ago, when he modified the existing UNIVAC version, implementing the domestic language vocabulary. Since that time, he has introduced many improvements to STRESS, including the dynamic analysis capabilities.

Nikolaj Ivancic is a mathematician, with a special love for computers. After many years of using large mainframes like CDC 6000, UNIVAC 1100 series, he was the first one around who discovered the power of micro. As a result of this AMS has been founded, joining many first class computer specialists from various fields together. Mr. Ivancic works in the same department as Dr. Nardini, with the same fields of interest — he implemented the well known structural analysis program SAP on various computers, including Cromemco CS-3.

Miljenko Srkoc is a civil engineer, and besides lecturing at the Structural Mechanics Department, he is the man who uses programs like STRESS and SAP in real applications. In the last 10 years he has been involved in most of the department's large projects, which used structural analysis programs as a tool.

Cromemco Introduces

SUDS

SOFTWARE UPDATE SERVICE

KEEPS REGISTERED SOFTWARE CURRENT

A January, 1981, announcement from Cromemco, Inc. may make the users of the company's software packages very happy, and avoid much of the confusion which has existed in the past.

Called SUDS, for Software Up-Date Service, the new subscription program works like this. Each time there is a production update of any software package, subscribers to the service for that package will automatically receive a diskette which contains the new software.

The subscriber will also receive, with the diskette, a description of how the new software differs from the previous version. When new documentation is required to use the updated version, it will be included in the mailing.

SUDS is available for a fixed annual fee, regardless of how many production updates are made during the year. The fee is a modest $95 per software package.

Registered users of Cromemco software have been sent pre-printed order forms showing those packages for which they are now registered, and for which the update service is provided.

A list of the current versions of all Cromemco software follows for use as a guideline.

If past performance is any indication of what is to come, the service can be very economical. For example, IACU has the $95 DBMS package which was issued prior to DBR. We have just sent our check for $95 for the new, combined DBMS/DBR valued at $295. We have already made $105 on the subscription service and we will get all updates for a full year. With the probability of improvements in this package during the coming year, we feel it is most profitable for us to take advantage of SUDS.

Current Versions of Cromemco Software

In response to many requests, Cromemco has supplied us with a complete list of the current versions of all its disk software products. Please check this list against the version of Cromemco software you are using, and use it as a guideline for determining whether you need to update.

The following sections list all Cromemco disk software, the packages in which they're sold, and their current verion numbers. Cromemco version numbers consist of four digits in two groups of two. The two digits before the decimal point represent the level of enhancement. This number is changed when significant new features are added or extensive changes are made to the software.

The two digits after the decimal point represent the version number of that particular level. This number is changed whenever anything is changed in the software, however slight. For example, if an error message in a program contains a misspelled word and this is subsequently corrected, the version number will be incremented by one. Generally, a product that has changed will have its version or revision numbers incremented by more than one. This does **not** mean that the customer has missed a version, merely that the software corresponding to the intermediate version numbers has remained internal to Cromemco.

Operating System Software

The following operating system software is supplied with all products **except** those which use an operating system other than CDOS: Multi-User Basic (model FDM and MUB) and Cromix Operating System (model CROMIX).

Program	Vers. Revs.
Cromemco Disk Operating System	02.36
CDOS	02.36
CDOSGENerator	02.36
Utilities	
@ (Batch)	02.00
DUMP (dump binary files)	00.07
EDIT (character-oriented text editor)	00.10
INIT (initialize disks)	02.15
SCREEN (screen-oriented text editor)	01.24
STAT (display system status)	02.16
WRTSYS (write system area)	02.00
XFER (transfer files)	01.07

System Software Products

Product Name & Primary Programs	Model	Vers.Revs.
Fortran IV	FDF	
FORTRAN compiler & library		03.37
LINKer		03.37
16K Extended Basic	FDB	
BASIC interpreter		05.70
Z80 Macro Assembler	FDA	
ASMB		03.07
DEBUGger		00.17
LINKer		03.21
* Trace System Simulator	TSS	02.06
TRACE simulator		02.06
DEBUGger		00.17
Dazzler Graphics	DGR	
Fortran & Basic graphics libraries		00.09
Word Processing System	WPS	
FMT (Formatter II)		06.00
SCREEN editor		01.24
Cobol	FDC	
COBOL compiler, overlays, & library		04.01
LINKer		03.37
Data Base Management System	DBM	
DBMS		03.05
DBR (data base reporter)		01.00
* Multi-User Basic	FDM	
OS using 32K of RAM in bank 7		01.52
OS using 16K of RAM in bank 7		01.42
Multi-User BASic interpreter		01.01
32K Structured Basic	STB	
SBASIC interpreter		03.65
BASICGENerator		03.65
LIBBUILD (procedure-library builder)		00.10
Ratfor with Fortran IV	FDR	
RATFOR preprocessor		01.00
FORTRAN compiler & library		03.37
LINKer		03.37
RPG II Business Language	RPG	
RPG compiler & library		01.02
LINKer		03.37
RPGEDIT (RPG editor)		01.94
REFM (IBM-CDOS reformatter)		01.03
SDI Graphics Software	SGS	
Fortran graphics library		01.05
Basic graphics library		00.05
PIXSAVE (SDI picture compression program)		01.02
PIXLOAD (SDI picture decompression program)		01.03
System Diagnostics Software	CDS	
HDIAG (hard disk diagnoser)		00.09
DISKDIAG (floppy disk diagnoser)		00.09
LISP	LSP	01.07
Cromix Operating System	CROMIX	10.09

*** Denotes Software not eligible for SUDS**

Join Now. But Don't Cut The Page. Copy This Application and Mail to IACU With Your Check.

Application for Membership J/F 81

Please start my Membership in the International Association of Cromemco Users right away. I have enclosed my
( ) Check ( ) Money Order in the amount of

$ ________________ for a ( ) year membership.
(U.S. Dollars only please)

Membership Rates in the United States:
( ) 1 yr. = $35.00 ( ) 2 yr. = $65.00
( ) 3 yr. = $90.00

Membership Rates in Canada and Mexico:
( ) 1 yr. = $41.00 ( ) 2 yr. = $77.00
( ) 3 yr. = $108.00

Membership Rates in all other countries:
( ) 1 yr. = $48.00 ( ) 2 yr. = $81.00
( ) 3 yr. = $129.00

Name: ________________

Title: ________________

Company: ________________

Mailing Address: ________________

City: ________________ State: ________

Country: ________________ Zip: ________

Phone: ________ ________ ________
area code number extension

Mail Your Membership Application to:
The International Association of Cromemco Users
P.O. Box 17658, Irvine, CA 92713
For more information, call (714) 955-0432

The International Association of Cromemco Users is designed to provide its Members with the information they want. Help us deliver by answering the following questions. You may check more than one block as applicable: My field is:

( ) Professional Services (Accounting, Dentist, Law, Medicine, Pharmacy, other: ________________)
( ) Engineer: ________________
(indicate specialty)
( ) Retail Business
( ) Wholesaler or Distributor
( ) Educational Institution: ________________
(indicate level)
( ) Government: ________________
(indicate branch)
( ) Home Use
( ) Other: ________________
(describe)

32

My Primary Uses are:
( ) Business
( ) Business & Personal
( ) Educational
( ) OEM
( ) Personal Only
( ) Process Control
( ) Other: ________________

I am Interested in:
( ) Firmware
( ) Hardware
( ) Software
( ) Other: ________________

I Want to Know More About the following Packages:
( ) Accounting
( ) Educational (adult)
( ) Educational (child)
( ) Computer Graphics
( ) Inventory Control
( ) Investments
( ) Home Economics
( ) Medical Research
( ) Process Control
( ) Sports & Games
( ) Other: ________________

Describe Your Present System:
(use brands and model numbers)
Computer: ________________
Memory: ________________
I/O: ________________
Disk: ________________
Terminals: ________________
Printers: ________________
Other: ________________

List Software Now in Use:
(Packages, Special Operating Systems, etc.)

____

What Types of Software Would You Like to know more about?

____

Would you be interested in preparing an article of interest to our members?

☐ Yes

Subject Matter: ________________

Cromemco Leads With New C Compiler

Continued from first page

programmer is also allowed to define new types using the **typedef** statement. C contains a very large selection of operators (40), and allows the programmer to define values of symbols and to use C language macros with macro arguments. A C source file can contain a directive which causes the compiler to insert the contents of another source file during compilation (file inclusion), and a C program can be formed by combining one or more separately-compiled modules. Cromemco C also contains an assembly-language window feature, that is, a C program can contain blocks of in-line assembly code.

The Cromemco C compiler executes only under the CROMIX operating system, but C programs may be compiled and linked to execute under either CDOS or the CROMIX operating system. The CROMIX operating system requires a minimum of 128K of memory, and C users need at least 243K of disk space for all of the programs and files supplied with the C package. The C package contains two C function libraries. One contains functions which use CDOS system calls, the other contains functions which use the faster CROMIX calls. All programs which execute under CDOS will also execute under the CDOS simulator on CROMIX system.

The Cromemco C compiler operation consists of three passes, or phases, a more accurate term since C reads the original file only once. The three phases read the C source program and produce an assembly language file which contains the assembly language code to perform all of the I/O, logic, and data manipulation expressed by the C programmer. The assembly language file is input to the Cromemco Macro Assembler, which is supplied with the C package. The assembler generates a relocatable object file which is combined with any separately-compiled program modules and any necessary functions from the C library by the Link utility program to create an executable object file.

The Cromemco C compiler compiles at a rate of 146 lines per minute on the HDD hard disk drive. This figure does not include assembling and linking time. When the assembling and linking times are included, the rate is 100 lines per minute.

One of the best testimonials to Cromemco C is that the Cromemco R & D Department itself is now using the language extensively for the development of new software products. The DBR report writer, recently released by Cromemco, is one example of a product written entirely in C. And, interestingly, the C compiler itself was written in C!

The C compiler is now available from Cromemco dealers on either 5" or 8" double-sided, double-density diskettes formatted for use with the CROMIX operating system. (For more information about this product, be sure to read the article by Dave Ellis in this issue.)

String Arrays

Continued from first page

sion string "A$" (99 10 byte segments).

Another example:

10 Print A$(60,72)

In Cromemco BASIC this command will print the contents of the string "A$" from positions 60 through 72. In a string array, it would print the contents of the string "A$" at column 72, row 60.

Now suppose we wanted to write a program that would read 10 records from a file into "A$". The programs could look as follows:

Cromemco BASIC

```
10 Dim A$(I*J)
20 Rem I= number of records
30 Rem J= record length
40 Open\1,J\"File"
50 For X=0 to I-1
60 Get\1,X\A$(X*J)
70 Next X
```

Other BASIC

```
10 Dim A$(I,J)
20 Rem I= number of records
30 Rem J= record length
40 Open\1,J\"File"
50 For X=1 to I
60 Get\1,X\A$(X)
70 Next X
```

As you can see, these programs are quite similar in syntax, yet there are some subtle differences. The important differences are on lines 10 and 60. Line 10 dimensions essentially the same amount of space in each example. The Cromemco BASIC accomplishes this by multiplying the number of records by the record length (number of characters per record). In the other BASIC, the string is dimensioned as per the first example. In line 60, the record is read into the string each time in a different segment of the string. In Cromemco BASIC, this is accomplished by multiplying the record counter "X" by the record length to determine the position within the string to place the record. In the other BASIC, this is accomplished by using the record counter "X" within the string itself.

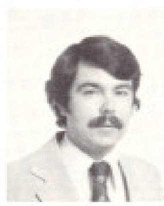
From these examples, it is evident that the implementation of programs using strings in Cromemco BASIC is actually quite easy to do.

A further feature found in Cromemco 32K Structured BASIC, is that the programmer can dynamically expand the size of a string as needed by using the EXPAND function. This allows the programmer to dynamically expand the size of the string in order to insert new characters.

For example:

```
10 Dim A$(14)
20 A$(-1)="A"+A$(-1) : Rem
     This fills A$ with "A"'s
30 Expand A$(8), 4
40 A$(8,-4)="BBBB"
50 Print A$
       Run
     AAAAAAAABBBBAAA
```

In summary, the string function found in Cromemco BASIC is not a string array. The string function in Cromemco BASIC, however, can easily be used to implement code that might be implemented by string arrays in other languages, and offers additional advantages over these other versions.



About the Author

Christopher B. Rook is manager of the Technical Sales Department at Cromemco. He has been with Cromemco since 1977 and has worked with Cromemco products for 5 years. He holds a B.S.C.S. from UCSD and is currently an M.B.A. candidate at the University of Santa Clara.

COMMERCIAL MEMBERS

ALASKA SYSTEMS
2900 W. Northern Lights Blvd.
Anchorage, AK 99503
(907) 248-2700

Full service company, providing consultation. Stocks a large inventory of both hardware and software; specializes in payrolls, power generation monitor, custom device drivers. Provides warranty repair service.

Key Personnel: William Hutchison, Owner
R. Pereira, software specialist
R. Yorgenson, software specialist

Major Market Area:
Sales & Service: throughout Alaska

AGRO MARKETING
B. Adzije 7/I, 41000 Zagreb
Yugoslavia
41 417-662 Telex: 2141yuam

Large full-service facility, with complete line of Cromemco products and proprietary software. Specializing in software development, interfacing, and special medical computerized equipment.

Key Personnel: T. Raguz, Director (Marketing)
N. Ivancic, Software Manager
B. Krtolica, Customer Support (Hardware)

Major Market Area:
Sales & Service: Internationally, primarily Yugoslavia

AMERICAN COMPUTERS & ENGINEERS, INC.
2001 S. Barrington, Suite 204
Los Angeles, CA 90025
(213) 477-6751

Consulting engineers, sales and service. Provides on site or in-house service agreements to all Cromemco Users. Offers structural engineering programs, accounting and word processing.

Key Personnel: Ghassan Dib, Pres. (Ph.D. Structural Engineering)
Aziz Al-Khal, Industrial Engineer, (Sales & Marketing)
Marwan Dib, Mechanical Engineer, (General Manager)

Major Market Area:
Sales & Service: Stores in Berkeley, & Los Angeles, California, and Paris, France.

COLLINGSWOOD COMPUTER CENTER
2311 W. Route 70
Cherry Hill, NJ 08002
(609) 488-1144

Medium-sized software house, specializing in small business systems; all models of Cromemco/payroll, billing, mass mailer. Provides warranty service also contract and hourly service.

Key Personnel: Jim Lenz, Pres. (software design & development)
Eric J. Watkins, Sales Representative
Jim Masterson, Vice Pres. (service)

Major Market Area: Sales: New York to Washington, extending to entire U.S. Service: Metro Philadelphia extending to Eastern Corridor.

COMPUTER CENTERS OF AMERICA
2129 Westheimer Road
Houston, TX 77098
(713) 527-8008
2629 Stemmons Fwy., Ste. 215
Dallas, TX 75207
(214) 638-4477

Complete line of hardware and software in inventory, 10,000 square feet of space. Provides warranty repair service.

Key Personnel: Avery More, Pres. (sales)
Richard Hernlund, Dallas Manager
Ron Blake, Houston Manager

Major Market Area: Sales & Service: Houston and Dallas, Texas.

COMPUTERLAND
16 A Avenue Marnix
1050 Brussels
Belgium
2/54.34.45 Telex: 62973

Computer store, providing warranty repair service. Complete line of Cromemco equipment.

Key Personnel: T. Beyerman, Chairman (software)
J.P. Deville, Technician (hardware)

Major Market Area:
Sales & Service: Internationally, primarily Belgium

COMPUTER PRODUCTS INTERNATIONAL
3225 Danny Pkwy.
Metairie, LA 70002
(504) 455-5330

Software house, specializing in general business and wholesale distributor systems. Full line of Cromemcos, in 10,000 sq. ft. two story building.

Key Personnel: Dan Ellis, Pres.
Jim Miller, Vice Pres. (systems)
Ben Rauch, Vice Pres. (systems)

Major Market Area: Sales: Entire Gulf Coast. Extended area, U.S.

CUSTOM COMPUTER SPECIALISTS, INC.
208 Roanoke Avenue
Riverhead, NY 11952
(516) 369-2199

Full service systems house with retail showroom. Full line of Cromemco hardware, software, accessories, and literature. Provides warranty service, diagnostics, consultation, systems analysis, and custom programming. Special management software for attorneys, mass transportation scheduling, reservations, delivery manifests, education, small businesses. School rentals, teacher training.

Key Personnel: Gregory G. Galdi, Pres.

Continued on next page

Continued from page 23

Major Market Area: Sales: Northeast U.S., extending to East Coast. Service: East Coast extending to Continental U.S.

DATAX DE MEXICO SA
Saturnino Herran 77
Mexico 19, D F.
593-22-42

Computer store, with showroom displaying complete line of Cromemco equipment. Special emphasis on small businesses and in the educational field. Warranty repair service available.

Key Personnel: Ignacio Palomar, Gen. Mgr. (hardware)
Jose H. Espinosa, Program Mgr. (software)

Major Market Area:
Sales & Service: Internationally, primarily Mexico

DIGIBYTE SYSTEMS CORP.
31 East 31st Street
New York, NY 10016
(212) 889-8130

Computer store carrying a full line of Cromemco hardware and software. Warranty service is provided.

Key Personnel: Robert Silverman, Pres. (software)
Barry Becker, Vice Pres. (hardware)

Major Market Area:
Sales & Service: Primarily East Coast

DIGITAL TECHNOLOGY
10 North Third
Lafayette, IN 47901
(317) 423-2548

3,000 sq. ft. computer showroom, specializing in educational/business systems. Large inventory of Cromemco software. Warranty repair service available.

Key Personnel: Greg Madden, Proprietor (hardware)
Tim Stockman (software)

Major Market Area:
Sales & Service: Indiana, throughout Midwest

DIGITRONIC
Digitronic Computersystems GmbH
Am Kamp 17 . 2081 Holm
Holstein, West Germany
0 41 03 / 8 86 72 / 3 Telex: 02 189 561

Full service computer store, providing consultation and warranty repair service. Complete inventory of Cromemco boards, systems, software packages.

Key Personnel: Claus Martens
Holger Vorbeck
Peter Adebahr

Major Market Area:
Sales & Service: West Germany

DIGITUS LIMITED
9 Macklin Street
Coven Garden, London WC2
01-4056761 Telex - Ref: 3005

Complete line of Cromemco hardware and software, 4,000 sq. ft. of space. Specializing in training, management sciences, personnel systems. Provides consultation services.

Key Personnel: A.C. Wood, Managing Dir. (consultancy)
P.S. Woolfonden, Systems Mgr. (software)
S. Parel, Sales Mgr. (hardware)

Major Market Area: Sales & Service: London, extending throughout United Kingdom.

INFORMATIVE SYSTEMS P/L
3 Bank Street
South Melbourne, Victoria, Australia
03-6902284 TWX 30458

Full range of Cromemco, retail and wholesale computer store. Provides full sales and service, specializing in education and small business applications.

Key Personnel: Dr. Simon Rosenbaum, Mng. Dir.
Ian Savicky, Tech. Advisor
Norman Rosenbaum, Sales Mgr.

Major Market Area: Sales & Service: throughout Australia

INFOSOFT SYSTEMS, INC.
25 Sylvan Road South
Westport, CT 06880
(203) 226-8937

Supplier of sophisticated software to systems users and retailers. Complete line of Cromemco software and applications packages in stock, as well as hardware. Special interconnects to accommodate Cromemco software.

Key Personnel: Ken Short, Pres. (sr. programmer/ analyst)
Richard Roth, Vice Pres. Mktg. & Production
Peggy Herlihy, Customer Service (analyst)

Major Market Area: Sales: International. Service: U.S.

MCM ENTERPRISES
459 Hamilton Ave., #304
Palo Alto, CA 94301
(415) 493-3333

A full service computer solutions company with consulting, equipment, software, training, and service. MCM carries a full line of Cromemco Systems, Serendipity and Computer Information Systems software, and NEC Printing Terminals. Authorized NEC Service Center for Northern California (printers & printing terminals).

Key Personnel: M.C. Merchant, Owner (systems design)
G. Nielsen, Svc. Engr. (maintenance)
L. Yori, Mgr., Reno Office (systems design)

Major Market Area: Sales: San Francisco Peninsula & Nevada extending internationally. Service: S.F. Peninsula, Nevada, extending into Northern Calif.

Reno Office: 1275 Kleppe Lane, #14, Sparks, Nevada 89431 (702) 358-0415

LENDAC DATA SYSTEMS, LTD.
8 Dawson Street
Dublin 2, Ireland

Suppliers and supporters of the full range of Cromemco Computer Systems and software.

Key Personnel: Danny McNally, Director

Major Market Area:
Sales & Service: Throughout Ireland

LEAR DATA CORPORATION
3273 Claremont Way, Suite 203
Napa, CA 94558
(707) 252-7139

Systems House and full Cromemco dealership in professional, 3,000 square foot office facilities. Separate lab and repair facilities. 24-hour service responses. Provides full warranty service. Drive alignments done in-house.

Key Personnel: Robert Gustafson, Pres.
Dr. Joseph Nelson, Vice Pres.
Doug Sherrod, V.P./Mktg.

Major Market Area: Software - Nationwide
Hardware - Northern Calif.

MICAH (MICRO APPLICATIONS
& HARDWARE
73 Cazneau
Sausalito, CA 94965
(415) 332-4443

Software development and consulting house. Specializing in education, resorts, construction, and home health services. Special financing available for software development projects.

Key Personnel: Ninad Freedman, Pres. (Software Engineer)
Dale Maxwell, Sales Mgr.

Major Market Area: Sales: International. Service: San Francisco Bay Area & Western U.S.

MICROCENTRE LIMITED
30 Dundas Street
Edinburgh EH3 6JN Scotland
031-556 7354 Telex: 72165 Ref. W582

Complete line of Cromemco Catalog items in inventory. 8,000 sq. ft. computer warehouse, extensive workshop for testing of equipment. Warranty repair service is available.

Key Personnel: Norman Rouxel, B.SC., Director (Cromemco equipment)
Andrew Smith, B.SC., Director (Cromemco equipment)
John Pringle, B.SC., Director (Business)

Major Market Area: Sales & Service: Entire United Kingdom, extending to North Sea Basin/Ireland

SOPORTE ADMINISTRATIVO
COMPUTACIONAL, S.A.
15 De Mayo #1111 PTE
Monterrey, N.L. Mexico
43-83-40

Complete line of Cromemco hardware and software in inventory. Specializing in the educational field. Full service facility, providing technical consulting, as well as warranty repair service.

Key Personnel: Juan Angel Perez, Director (systems)
Jaime Martinez, Customer Support (MSEE)
Gerardo Elizondo, Technical Mgr. (MSEE)

Major Market Area:
Sales & Service: Internationally, primarily Mexico

SYNERGISTICS INTERNATIONAL LTD.
35 Fountain Square Plaza, Box 631
Elgin, IL 60120
(312) 695-7775

Computer store and software house, with complete line of Cromemco equipment in inventory. Specializing in sales to small business turnkey systems.

Key Personnel: Jim Knowles, Pres. (Sales)
Gordon Muirhead, Vice Pres. (software)

Major Market Area: Sales: Chicago and suburbs, extending to entire U.S. and the United Kingdom. Service: Chicago and suburbs.

TRADEWIND SYSTEMS
Box 96
Liberal, KS 67901
(316) 624-8111, O/S KS 1-800-835-2057

Exclusive Cromemco dealer, specializing in complete business systems. Provides consulting services. Full inventory.

Key Personnel: Clark Stewart, Pres. (business systems)
Wayne Stewart, Vice Pres. (tech./software)
Kevin Elmore, Programmer/analysis

Major Market Area: Sales: S.W. Kansas, extending to COlorado, Kansas, Oklahoma, Texas, New Mexico. Service: S.W. Kansas.

XITAN SYSTEMS, LTD.
23 Cumberland Place
Southampton, England
0703 38740

Complete line of computers, 1,000 sq. ft. Also add-ons for Cromemco. All Cromemco software included in inventory. Provides warranty repair service.

Key Personnel: G.C. Lynch, Managing Dire. (sales & technical support)
R. Wilmott, Product Support Engineer (engineering)
J. Rosser, Personal Assistant (order processing)

Major Market Area: Providing U.K. & E.E.C. with total business, educational scientific, medical and industrial support.

Estimate Taxes and Save

by Theodore R. Johnson, Jr.

I. Summary and Objectives

The purposes of this paper are: 1) to indicate the importance to an investor of making periodic estimates of his current and future income tax liability, 2) to present a simple computer program in (Cromemco) BASIC which eases the task of making such estimates.

Although the objectives of the Micro Computer Investors Association (MCIA) are stated primarily in terms of negotiable securities (one does not often take delivery of commodities). I have taken the liberty of including in this paper references to partnership interests in such tax shelter investments as real estate and drilling programs, reflecting my belief that such investments are a necessary part of the portfolio of any investor who is concerned about the steadily increasing impact of inflation and taxes.

II. Why estimate taxes?

Although it hardly seems necessary to justify the desirability of minimizing income taxes, I am frequently surprised by the failure of investors of my acquaintance to take the time and trouble to make a quantitative evaluation of the tax implications of various courses of action. To be sure, an investor in the 70% bracket does not require elaborate calculations to appreciate the attractiveness of a promising tax shelter investment. Even he, however, may wish to make a rational determination of how much to invest in programs which, by their nature, entail considerable risk. A drilling program partnership interest purchased with 30cent dollars may seem irresistible, but the **risk:reward** ratio changes appreciably as one passes below the 50% bracket.

In other areas, the investor may have to decide whether (and when) to realize capital gains, whether it may be desirable to accelerate or defer income and/or deductions, whether to buy taxable or tax-free bonds, whether to buy an annuity, whether to seek an exchange or make an installment sale of an asset, how much to contribute to an IRA or Keogh plan, how to optimize distributions from a retirement plan or annuity, etc., etc. While it would be foolish to allow tax considerations to dominate one's investment decisions, it seems equally foolish to ignore them. And, in my experience, the effects of income averaging, minimum tax on tax preference, investment tax credits, and alternative minimum tax make "back of the envelope" calculations almost completely useless.

Other reasons to estimate taxes might be:

1) To avoid the loss of income resulting from excesive estimated tax payments (or over-withholding).

2) To avoid penalties and interest expense resulting from underpayment.

3) To anticipate the need to liquidate assets to make the final tax payment in April.

4) To provide additional justification for deducting depreciation and other outlays associated with your microcomputer as investment expenses.

5) To minimize the effort (and expense if professional assistance is utilized) of preparing the annual return.

As a guiding principle, one might select a particular tax bracket as a target, seeking to "normalize" his tax liability from one year to the next (since savings can always be realized by shifting income out of, and deductions into, a year of higher tax liability). Which bracket is selected depends upon the individual's personal and political philosophy, his appetite for risk, his willingness to be audited, etc.

III. When to Estimate Tax Liability

There are at least four occasions during the year when an estimate of federal income tax liability is quite useful. First, and least obvious, is at the very beginning of the taxable year. A preliminary and necessarily rough estimate will indicate whether to begin searching for tax shelter investments (the most attractive programs are generally offered during the first half of the year — to those investors who plan ahead) and will provide a target deduction amount. Needless to say, one must also estimate his cash flow (or liquidity) in light of, among other things, his tax liability for the preceding year, a large fraction of which may be due in April.

The second obvious occasion for tax estimating is in April to decide whether to base withholding and/or advance payments on last year's actual, or this year's estimated tax. At current interest rates this decision is not trivial and can result in significant additional income on funds which would otherwise find their way prematurely into the coffers of the Treasury Department. It goes without saying, of course, that if you use an estimate of this year's tax it should be checked frequently throughout the year to avoid possible underpayment and resulting penalty.

Thirdly, additional estimates should be made from time to time during the year as changes occur and more accurate information becomes available. As tax shelter investment opportunities arise, the actual tax impact can be evaluated quickly and easily to determine the real dollar cost of the investment. As stated earlier, the effects of averaging, the preference tax and the alternative minimum tax sometimes produce surprising results.

Finally, towards the end of the year estimates should be made of the current year's tax liability, reflecting the more accurate inputs which should have become available by that time. If one's tax liability, based on estimates for the current year, has increased, a penalty can be avoided by submitting a revised estimate and making a higher payment on January 15.

At this time an estimate should also be made of next year's tax to determine whether it would be advantageous to make adjustments between years. Most investors have more flexibility in this regard than they think,

and the payoff is not insignificant. A thousand dollars worth of deductions (property taxes, for example) switched to a year in which one enjoys a 10 percentage point higher marginal tax rate is worth $100.

IV. How to Estimate Tax Liability

Several tax computation programs are available in BASIC (the only computer language with which I am familiar). Most of them are designed for the preparation of the actual tax return (See Section VII) and, therefore, require a multiplicity of inputs which are not really necessary for the preparation of "quick and dirty" estimates.

The program presented herewith requires only 6K of memory and consists of five sections, each of which will be discussed briefly:

A. **Inputs** — Provision is made for inputting five items of ordinary income, longterm capital gains or losses and an IRA (or Keogh) deduction. "PSI" refers to "Personal Service Income", a bureaucratic euphemism for salaries, wages, consulting fees, and like items eligible for maximum tax treatment. Since the program was optimized for my particular situation, other users may wish to modify the input section to reflect their own circumstances. Active traders would probably find a "Short Term Capital Gain" input useful. Provided that the same numeric variables are used, no other changes in the program should be required. If a larger number of inputs is desired, appropriate changes should be made in LINE 400 which calculates Adjusted Gross Income.

Next, inputs are requested for itemized deductions and personal exemptions (both in total). Subtracting these from Adjusted Gross Income provides Taxable Income.

It is suggested that separate hand tabulations be maintained for estimates of each category of income and capital gains. Columnar paper is handy to provide a record of revisions made throughout the year. A similar approach is recommended to keep track of estimated deductions. Personal exemptions are now $1,000.

B. **Calculation of Scheduled Tax** — This section is basically a replication of the 1979 IRS tax table for married couples filing joint returns. Those in other categories will find it necessary to substitute the appropriate figures in LINES 1400 through 2030.

C. **Income Averaging** — This section is applicable to those who have an unusually high level of income for the current year. Specifically, savings will result if current taxable income is at least $3,000 in excess of 120% of the average taxable income for the four previous years. In calculating Base Period Income, taxable income originally reported for 1976 (and before) must be increased by $3,400 (for married taxpayers filing joint returns) to reflect the recent change in the tax schedule. If it is known that income averaging will not produce savings, an input of "0" to the prompt "BASE PERIOD INCOME" will cause this section of the program to be skipped.

D. **Credits and Preference Items** — Investment tax credits are available from some tax shelter partnership investments. Personal credits may be taken for the purchase of capital items (such as microcomputers) used for investment purposes. The amount of the credit depends on the period over which the item is depreciated. (Maximum of 10% for seven years or longer).

There are several so-called "tax preferences" to which a 15% add-on minimum tax applies if, in total, they exceed $10,000 or one-half of the regular tax liability, whichever is greater. The major preference items, which apply only to tax shelter investors, are 1) the excess of accelerated over straight-line depreciation on real property, and 2) intangible drilling costs on productive wells in excess of net oil and gas income.

An investor in a drilling or real estate partnership should, if possible, obtain from the General Partner estimates of credits and preferences along with the expected deduction from ordinary income. Since, in my experience, such estimates are difficult to come by, an educated guess is frequently the best one can do.

E. **Tax Calculations** — With these inputs the program proceeds to calculate "Preference Tax" (my label for the minimum tax on preference items) and "Regular Tax" (Scheduled Tax plus Preference Tax minus Credits and Savings for Averaging). For the calculation of Alternative Minimum Tax an additional input is requested: Exempt Deductions, State and Local Taxes, Medical Expense and Casualty Losses may be subtracted from Total Itemized Deductions before calculating the extent to which this amount exceeds 60% of Adjusted Gross Income. Considering the limitations on medical deductions and the infrequency of casualty losses, this input will normally be confined to state and local taxes. Adding excess deductions (if any) to the previously untaxed 60% of Capital Gains to determine "Alternative Minimum Taxable Income" (not printed out), the program calculates Alternative Minimum Tax.

Finally, it prints the Total Tax Due, requests an input of Estimated Tax Payments (or Withholding) and provides the Amount Payable in April.

V. Sample Runs

Three sample runs are provided. Case 1 depicts a reasonably prosperous investor with a broad spectrum of investment activity whose tax liability is nicely under control

```
Case 1
ESTIMATED FEDERAL INCOME TAX FOR? 1980
PSI? 40000
DIVIDENDS AND INTEREST? 7500
PARTNERSHIPS? -17000
COMMODITIES? 3500
OPTIONS? 2500
OTHER? 0
CAPITAL GAINS? 15000
IRA CONTRIBUTION? 1500
ADJUSTED GROSS INCOME =                          41000
ITEMIZED DEDUCTIONS? 8500
PERSONAL EXEMPTIONS? 2000
TAXABLE INCOME =                                 33900
SCHEDULED TAX (37% BRACKET) =                     7681
BASE PERIOD INCOME? 125000
SAVINGS FROM AVERAGING =                             0
INVESTMENT CREDIT? 500
PREFERENCE ITEMS? 12000
PREFERENCE TAX =                                   300
TOTAL REGULAR TAX =                               7481
EXEMPT DEDUCTIONS? 1500
ALTERNATE MINIMUM TAX =                            280
TOTAL TAX DUE =                                   7481
```

Continued on page 29

Estimate Taxes

Continued from page 27

ESTIMATED TAX PAYMENTS? 7000	
AMOUNT PAYABLE IN APRIL =	481

Although it is interesting to note that in Case 1, without the $17,000 partnership deduction he would be in the 49% bracket with a total estimated tax of $15,219.

In Case 2, he for some reason, realizes an additional capital gain of $135,000 which pushes him into the 59% bracket and increases his total tax liability by a factor of 4. It should be noted, however, that the effective tax rate on the additional gain is only about 17%, the tax being reduced by $3502 due to income averaging and by $300 due to a decrease in the minimum tax on preference items.

Case 2

ESTIMATED FEDERAL INCOME TAX FOR? 1980	
PSI? 40000	
DIVIDENDS AND INTEREST? 7500	
PARTNERSHIPS? −17000	
COMMODITIES? 3500	
OPTIONS? 2500	
OTHER? 0	
CAPITAL GAINS? 150000	
IRA CONTRIBUTION? 1500	
ADJUSTED GROSS INCOME =	95000
ITEMIZED DEDUCTIONS? 8500	
PERSONAL EXEMPTIONS? 2000	
TAXABLE INCOME =	87900
SCHEDULED TAX (59% BRACKET) =	34859
BASE PERIOD INCOME? 125000	
TAX FROM INCOME AVERAGING =	31357
SAVINGS FROM AVERAGING =	3502
INVESTMENT CREDIT? 500	
PREFERENCE ITEMS? 12000	
PREFERENCE TAX =	0
TOTAL REGULAR TAX =	30857
EXEMPT DEDUCTIONS? 1500	
ALTERNATE MINIMUM TAX =	18350
TOTAL TAX DUE =	30857
ESTIMATED TAX PAYMENTS? 7000	
AMOUNT PAYABLE IN APRIL =	23857

Since our hypothetical investor begrudges the IRS even this amount, he promptly investigates a 100% deductible drilling program which would raise his partnership deduction to $67,000. As shown in Case 3, this reduces his scheduled tax by $25,536, or 51¢ for every dollar invested. This reduction is offset, however, by the loss of his previously estimated $3502 savings from income averaging and a jump of $1800 in the additional minimum tax (due to an increase of $10,000 in estimated preference items), resulting in a total tax saving of only 35¢ on the dollar. This should certainly raise some doubt about the attractiveness of the investment.

Case 3

ESTIMATED FEDERAL INCOME TAX FOR? 1980	
PSI? 40000	
DIVIDENDS AND INTEREST? 7500	
PARTNERSHIPS? −67000	
COMMODITIES? 3500	
OPTIONS? 2500	
OTHER? 0	
CAPITAL GAINS? 150000	
IRA CONTRIBUTION? 1500	
ADJUSTED GROSS INCOME =	45000
ITEMIZED DEDUCTIONS? 8500	
PERSONAL EXEMPTIONS? 2000	
TAXABLE INCOME =	37900
SCHEDULED TAX (43% BRACKET) =	9323
BASE PERIOD INCOME? 125000	
TAX FROM INCOME AVERAGING =	9323
SAVINGS FROM AVERAGING =	0
INVESTMENT CREDIT? 1500	
PREFERENCE ITEMS? 22000	
PREFERENCE TAX =	1800
TOTAL REGULAR TAX =	9623
EXEMPT DEDUCTIONS? 1500	
ALTERNATE MINIMUM TAX =	13350
ADDITIONAL TAX =	3727
TOTAL TAX DUE =	13350
ESTIMATED TAX PAYMENTS? 7000	
AMOUNT PAYABLE IN APRIL =	6350

It should be emphasized that the numbers have been chosen in this rather unusual example primarily to illustrate some of the complexities of the tax calculation. In other, more typical circumstances the tax savings resulting from the higher deduction might be much greater.

VI. Limitations and Pitfalls

A number of observations about the program will be helpful in assessing its usefulness:

A. **Accuracy** — I have been estimating my income taxes with evolving versions of this program for three years during which time a number of programming errors have been corrected, new features have been added and changes have been made to reflect new tax regulations. After each of the last two years the program has been re-run with actual inputs from my federal tax return producing "estimates" within 1-2 percent of the actual tax liability. While this minimal effort at validation is too small to prove anything, it does provide some comfort to me.

It must be admitted, however, that the actual tax paid each year differed from the previous December's estimate by a much larger percentage. This is due to the fact that income and deductions, particularly those relating to tax shelter investments, cannot be estimated with precision. The chief limitation of the program, therefore, is the inherent inaccuracy of these inputs. As investment income increases in relation to earned income this problem becomes more significant, which I regard as a major incentive to use a computer program of the type presented here. Its major advantage is the relative ease by which one can monitor changes as they occur and make an advance determination of the tax impact of possible investment decisions.

B. **Omissions** — A number of complexities in the tax code have been deliberately avoided in the interest of simplicity. For example:

1) No provision is made for the 50% maximum tax on Personal Service Income. This is because my objective is to pay less than the 50% marginal tax rate, regardless of the source of my income. Those who are content to pay tax at the 50% rate and above will probably have little interest in estimating taxes anyway.

Continued on next page

Estimate Taxes

Continued from page 29

2) No provision is made for the earned income credit on the assumption that most investors with microcomputers have annual incomes above $10,000.

3) No provision has been made for the dividend exclusion simply because I have never been able to estimate dividends (combined with interest as an input) within $200 and I doubt whether most others could either. However, it would be easy to add.

4) No provision is made for estimating state income taxes because the primary purpose of the program is to provide a tax input to investment decisions and state tax effects normally follow those at the federal level in a much lesser amount. Adding a state tax schedule would be a relatively easy thing to do, however.

5) No provision is made for the depletion allowance on mineral assets nor for a number of other regulations dealing primarily with drilling programs. I have found that these can best be dealt with, if at all, in preparing the partnership income input.

6) No provision is made for the limitation on deductible investment interest (to $10,000 plus net investment income) simply because it has never been applicable to me. Those with a greater appetite for debt can, no doubt, make the appropriate additions to the program without difficulty.

7) For similar reasons no provision has been made for the limitation on deducting investment interest deemed to have been used to purchase or carry tax-free bonds.

8) In estimating alternative minimum tax, no provision is made for the exclusion of medical and casualty loss deductions from the amount to which this tax applies. If large deductions in these categories are anticipated, a separate calculation will be required.

9) Finally, as a programming tyro, I have not been able to devise a means of re-running the program to change a single input while preserving all others. This would add significantly to the convenience and utility of the program, and I would welcome suggestions to this end from my more knowledgeable brethren.

C. **General** — It is worth re-emphasizing that the program is intended to provide an estimate, not an exact computation, of income tax liability. Certain calculations, therefore, may represent approximations to those which will eventually be made on the return.

It is also worth mentioning that the tax regulations change frequently. (Significant changes have been made every year for the last five years, at least). Therefore, one must be willing to do an annual reprogramming job just to keep current. The program, as I have used it, is offered below.

```
100 @
110 @
120 @
130 @
140 @
150 DIM Y$(20)
160 @"ESTIMATED FEDERAL INCOME TAX FOR";
170 INPUT Y$
180 @
190 @
200 REM COMPUTE ADJUSTED GROSS INCOME
210 @"PSI";
220 INPUT I1
230 @"DIVIDENDS AND INTEREST";
240 INPUT I2
250 @"PARTNERSHIPS";
260 INPUT I3
270 @"COMMODITIES";
280 INPUT I4
290 @"OPTIONS";
300 INPUT I5
310 @"OTHER";
320 INPUT I6
330 @"CAPITAL GAINS";
340 INPUT G;
350 @"IRA CONTRIBUTION";
360 INPUT C
370 @
372 REM COMPUTE SHORT TERM GAIN OR LOSS
374 I7=I4+I5
376 IF G<0 THEN 390
378 REM LTG + STG
380 IF I7>0 THEN A=I1+I2+I3+I6+I7+(0.4*G)-C :
     GOTO 400
382 REM LTG + STL = NLTCG
384 IF I7+G>0 THEN
     A=I1+I2+I3+I6+(0.4*(G+I7))-C : GO TO 400
386 REM LTG + STL = NSTCL
388 A=I1+I2+I3+I6+I7+G-C : GO TO 398
390 REM LTL + STG = NSTCG
392 IF I7+G>0 THEN GOTO 388
394 REM LTL + STG = NLTCL, LTL + STL
396 A=I1+I2+I3+I6+(0.5*(I7+G))-C
398 IF I7+G<-3000 THEN A=A-(G+I7-3000)
400 IF A<0 THEN A=0
402 @USING"###############################
     ######","ADJUSTED GROSS INCOME", A
404 IF I7+G<-3000 THEN @"***YOU MAY HAVE
     A CAPITAL LOSS CARRYOVER TO NEXT
     YEAR***"
406 @
408 REM TO AVOID COMP OF NEGATIVE ALT MIN
     TAX
410 IF G<0 THEN G=0
430 REM COMPUTE TAXABLE INCOME
440 @"ITEMIZED DEDUCTIONS";
450 INPUT D
460 D=D-3400
470 IF D<3400 THEN D=3400
480 @"PERSONAL EXEMPTIONS";
490 INPUT E
500 @
510 T=A-D-E
520 @USING"###############################
     ######","TAXABLE INCOME=",T
540 @
550 REM COMPUTE SCHEDULED TAX
560 GOSUB 1400
570 Z=B
580 @USING"############## ## ############
     ######","SCHEDULED TAX(",P,
```

```
        "%BRACKET) = ",Z
590 @
600 REM COMPUTE SAVINGS FROM INCOME
        AVERAGING
610 Y=T
620 @"BASE PERIOD INCOME";
630 INPUT L
640 IF L=0 THEN 760
650 T=0.3*L
660 GOSUB 1400
670 R=B
680 T=0.3*L+0.2*(Y-0.3*L)
690 GOSUB 1400
700 S=B-R
710 IF S<=0 THEN 760
720 U=4*S+B
730 @USING"################################
        ######","TAX FROM INCOME AVERAGING = ",U
740 X=Z-U
750 IF X>0 THEN 770
760 X=0
770 @USING"################################
        ######","SAVINGS FROM AVERAGING = ",X
780 @
830 @"INVESTMENT CREDIT";
840 INPUT J
850 IF J<Z THEN 880
860 IF J>Z THEN J=Z
870 @"ADJUSTED CREDIT = "INT(J)
880 @"PREFERENCE ITEMS";
900 INPUT M
910 @
920 REM COMPUTE MINIMUM TAX ON PREFERENCE
        ITEMS
930 IF(Z-X-J)/2<10000.0 THEN 960
940 H=0.15*(M-(Z-X-J)/2)
950 GO TO 970
960 H=0.15*(M-10000.0)
970 IF H<0 THEN H=0
1000 @USING"################################
        ######","PREFERENCE TAX = ",H
1010 @
1020 V=Z+H-J-X
1030 IF V<0 THEN V=0
1040 @USING"################################
        ######","TOTAL REGULAR TAX = ",V
1050 @
1060 REM COMPUTE ALTERNATIVE MINIMUM TAX
1070 @"EXEMPT DEDUCTIONS";
1080 REM STATE & LOCAL TAXES, MEDICAL
        EXPENSES, CASUALTY LOSSES
1090 INPUT Q
1100 @
1110 REM COMPUTE EXCESS DEDUCTIONS
1120 N=A-(D+3400+E)+(D+3400-Q)-0.6*(A-Q)
1130 IF N<0 THEN N=0
1140 REM COMPUTE ALTERNATIVE MINIMUM
        TAXABLE INCOME
1150 O = N + 0.6*G
1160 IF O<20000.0 THEN 1210
1170 IF O>100000.0 THEN 1230
1180 IF O>60000.0 THEN 1250
1190 F=0.1*(O-20000.0)
1200 GOTO 1260
1210 F=0
1220 GOTO 1260
1230 F=12000.0+0.25*(O-100000.0)
1240 GOTO 1260
1250 F=4000+0.2*(O-60000.0)
1260 @USING"################################
        ######","ALTERNATE MINIMUM TAX = ",F
1270 @
1280 IF F<V THEN 1330
1290 W=F-V
1300 IF W<0 THEN W=0
1310 @USING"################################
        ######","ADDITIONAL TAX = ",W
1320 @
1330 @USING"################################
        ######","TOTAL TAX DUE = ",V+W
1340 @
1350 @"ESTIMATED TAX PAYMENTS";
1360 INPUT K
1370 @
1380 @ USING"################################
        ######","AMOUNT PAYABLE IN APRIL = ",
        V+W-K
1390 GOTO 2060
1400 IF T>215400.0 THEN 1560
1410 IF T>162400.0 THEN 1590
1420 IF T>109400.0 THEN 1620
1430 IF T>85600.0 THEN 1650
1440 IF T>60000.0 THEN 1680
1450 IF T>45800.0 THEN 1710
1460 IF T>35200.0 THEN 1740
1470 IF T>29900.0 THEN 1770
1480 IF T>24600.0 THEN 1800
1490 IF T>20200.0 THEN 1830
1500 IF T>16000.0 THEN 1860
1510 IF T>11900.0 THEN 1890
1520 IF T>7600 THEN 1920
1530 IF T>5500 THEN 1950
1540 IF T>3400 THEN 1980
1550 IF T<=3400 THEN 2010
1560 B=117504.0+0.7*(T-215400.0)
1570 P=70
1580 RETURN
1590 B=81464.0+0.68*(T-162400.0)
1600 P=68
1610 RETURN
1620 B=47544.0+0.64*(T-109400.0)
1630 P=64
1640 RETURN
1650 B=33502.0+0.59*(T-85600.0)
1660 P=59
1670 RETURN
1680 B=19678.0+0.54*(T-60000.0)
1690 P=54
1700 RETURN
1710 B=12720.0+0.49*(T-45800.0)
1720 P=49
1730 RETURN
```

Continued from page 31

```
1740 B=8162+0.43*(T-35200.0)
1750 P=43
1760 RETURN
1770 B=6201+0.37*(T-29900.0)
1780 P=37
1790 RETURN
1800 B=4505+0.32*(T-24600.0)
1810 P=32
1820 RETURN
1830 B=3273+0.28*(T-20200.0)
1840 P=28
1850 RETURN
1860 B=2265+0.24*(T-16000.0)
1870 P=24
1880 RETURN
1890 B=1404+0.21*(T-11900.0)
1900 P=21
1910 RETURN
1920 B=630+0.18*(T-7600)
1930 P=18
1940 RETURN
1950 B=294+0.16*(T-5500)
1960 P=16
1970 RETURN
1980 B=0.14*(T-3400)
1990 P=14
2000 RETURN
2010 B=0
2020 P=0
2030 RETURN
2060 @
2070 @
2080 END
```

VII. Bibliography

A. **Books** — Numerous volumes on the general subject of taxes are available ranging from texts for college courses in tax preparation to the latest expose by a "former IRS employee." One which I have found useful is "Tax Planning for Investors" by Jack Crestol and Herman M. Schneider, Dow Jones Books, Princeton, New Jersey. It appears to be updated annually or at least often enough to keep up with major changes in tax law.

B. **Periodicals** — Two sources of current tax information can be recommended. First, the Tax Report, which appears each Wednesday in the Wall Street Journal, is probably the best way of keeping up with IRS regulations and rulings and Tax Court decisions which may affect one's immediate interests. Second, a monthly newsletter called "Tax Angles" is published by Kephart Communications, Inc., 901 N. Washington Street, Suite 605, Alexandria, VA 22314, at a current subscription rate of $44/year. In addition to tax matters relating to investors it contains articles aimed at professionals and small employers.

C. **Computer Programs** — I have seen several programs advertised for the preparation of actual tax schedules. Having no experience I am unable to comment on their value. A recent issue of Creative Computing contains an ad by Gooth Software, 931 S. Bemiston, St. Louis, MO 63105 for a "Tax Program Book" which offers such program listings plus "helpful programming hints" for $14.95.

Two articles in microcomputer-oriented periodicals may be of interest. "Important Lessons You Can Learn from Estimating Your Federal Income Taxes" by W.A. Tinsley in the February, 1980 issue of Creative Computing (Vol. 6, No. 2, p. 54) focuses on such questions as whether the taxpayer will receive a refund or have to pay more taxes at the end of the year. The article does not identify the language in which the program is written (it is unfamiliar to me). No sample output is provided.

A "Tax Calculation Program" by Gary O. Young appears in the January, 1978 issue of Interface Age (Vol. 3, Issue 1). The program, which is quite lengthy (about 375 lines) is written in Northstar DOS BASIC. The sample run provided is obviously incomplete, but the program appears to be a more detailed and comprehensive effort to achieve objectives similar to mine. The program includes Federal and California State tax tables for single taxpayers (since revised). It also includes provision for FICA and detailed inputing of specific deductions and chooses between itemized and standard deductions, but appears to omit preference items and the alternative minimum tax (which, of course, had not been invented then). For those readers desirous of creating a customized program of their own, it is well worth reading.

About the Author:
Theodore R. Johnson, Jr.

Ted Johnson is a retired business executive living in Inverness, California who took up programming both as a hobby and as an aid in monitoring his investments. He developed his income tax estimating program to fill a need which is not satisfied by existing software designed to prepare actual tax returns. Although he is not a CPA and has no professional standing in either the tax or investment field he thinks that the program may be useful to those who wish to take income taxes into account in making investment decisions.

bits & bytes, nibbles & tweaks...

SOUTH JERSEY & PHILLY USER'S GROUP UP AND RUNNING

The organization meeting of the South Jersey & Philly Cromemco User's Group (announced in I/O News, Vol. I, No. 2) was scheduled for Feb. 25th. The first part of the meeting was to be devoted to the structure of the group, and the second half to the specific interests of the members. There is a new contact and phone number for those wanting to become part of the group. Eric Watkins, at (609) 488-1144.

WALNUT CREEK USER'S GROUP

The February meeting of the Walnut Creek (California) User's Group featured a presentation by Hal Nissley from Condor Systems. Hal introduced his Relational Data Base which allows the integration of several types of records into one reporting scheme. Now we'd like some feedback as to what the members thought of the package. Those interested in affiliating with the group may call Hank Couden at (415) 935-6502.

NEW GROUP IN NORTHWEST

NWACU, the Northwest Association of Cromemco Users, has been formed with its next meeting scheduled for March 30th. Meetings are regularly scheduled for the last Monday of each month at 7:30 PM at:

Maverick Microsystems
14808 NE 31st Circle
Bldg. A
Koll Business Center
Redmond, Wa

NWACU bills itself as being affiliated with the Northwest Computer Society and the IACU. For more information, contact: Jim Illman at (206) 932-8771.

INCOME TAX PREPARATION SOFTWARE

A member in San Diego writes: "We have a hard working Cromemco that is the backbone of our various company services. We would very much like to add Income Tax Preparation to our portfolio . . . Is there such a program among your associates? If so, kindly advise how we may participate . . ." Okay, Computer Tax specialists out there. Can anyone help out with a program for preparing income taxes? Contact IACU.

ARTICLE TO BE PREPARED IF ENOUGH MEMBERS INTERESTED

We received this generous offer from a member in Northridge, California:

"The terminal I am using with my Cromemco System Three has 32 special function keys; (actually 16 keys,, each performing a second function when used with the shift key). CDOSGEN.COM and CDOS.COM version 2.17 will only support a maximum of 20 special functions."

"Using the source file and information on CDOS I/O DRIVERS furnished by Cromemco, I was able to modify the drivers to support the extra function keys . . . I wrote a program in SBASIC to insert the function key address and definitions into the CDOS I/O DRIVERS."

If there are enough members who have this problem . . . I will write an article on it."

Are there? Let us know.

SASKATCHEWANIANS NEED SOFTWARE

From Regina, Saskatchewan came this very specific request:

bits & bytes

Continued from page 33

"We are looking for a Student Registry or Student Information System, preferably written in COBOL. Any leads would be appreciated."

Can you help with this request? Please contact IACU if you know of any applicable software.

LAW OFFICE NEEDS BILLING PROGRAM

A small law office in Northern Illinois would like to acquire inexpensive software which will allow them to use their System Three for time-keeping and billing. They would also like to see a demonstration of a multi-user word processor in operation. Any volunteers? Contact IACU.

COMPENSATION + SOFTWARE EXCHANGE

A member in Arnprior, Ontario (Canada) has suggested we find some way of compensating contributors for their editorial material.

Gee, we would really like to do that. The problem is, we simply can't at this time. However, it may please you to know that many of the articles being submitted are by people with an economic interest in the subject matter. And several have found that their articles in I/O News have resulted in more new business than they ever imagined. Give us a year or two of history, and we may devise some form of compensation.

The same member also sent us a few tips as to what DECUS (the DEC Users' Group) is doing in the area of software exchange. That is a program we will look into in detail and report to you. Let us know how you feel about this.

WEST LOS ANGELES GROUP UP AND RUNNING

Last issue we printed a request from a member in Santa Monica who expressed interest in meeting with other Cromemco users. The results were sufficient to create a local group — as yet unnamed — which is holding its first meeting on Feb. 24th. We have been invited to this meeting and will report any news next issue.

TIPS ON CDOS DISK CALLS

The following letter from Jim Gunkel of Beavercreek, Ohio could be of use to many members, so we decided to reprint it. It is the type of information we enjoy receiving.

Editor - I/O News:

I have come across an error in one of Cromemco's manuals that may give people using Assembly language or converting CPM software some problems. The following is quoted for reference -

CROMEMCO DISK OPERATING SYSTEM (CDOS) User's Manual
Part No. 023-0036 February 1980
Series-2 CDOS Instruction Manual
paragraph II.1.5 on pages 6 and 7 of supplemental information

second paragraph ...The directory always begins with Sector 1 of a particular track, ...

third paragraph As stated, Sector 1 of these tracks begins the directory and is the start of the sector interleaving ...

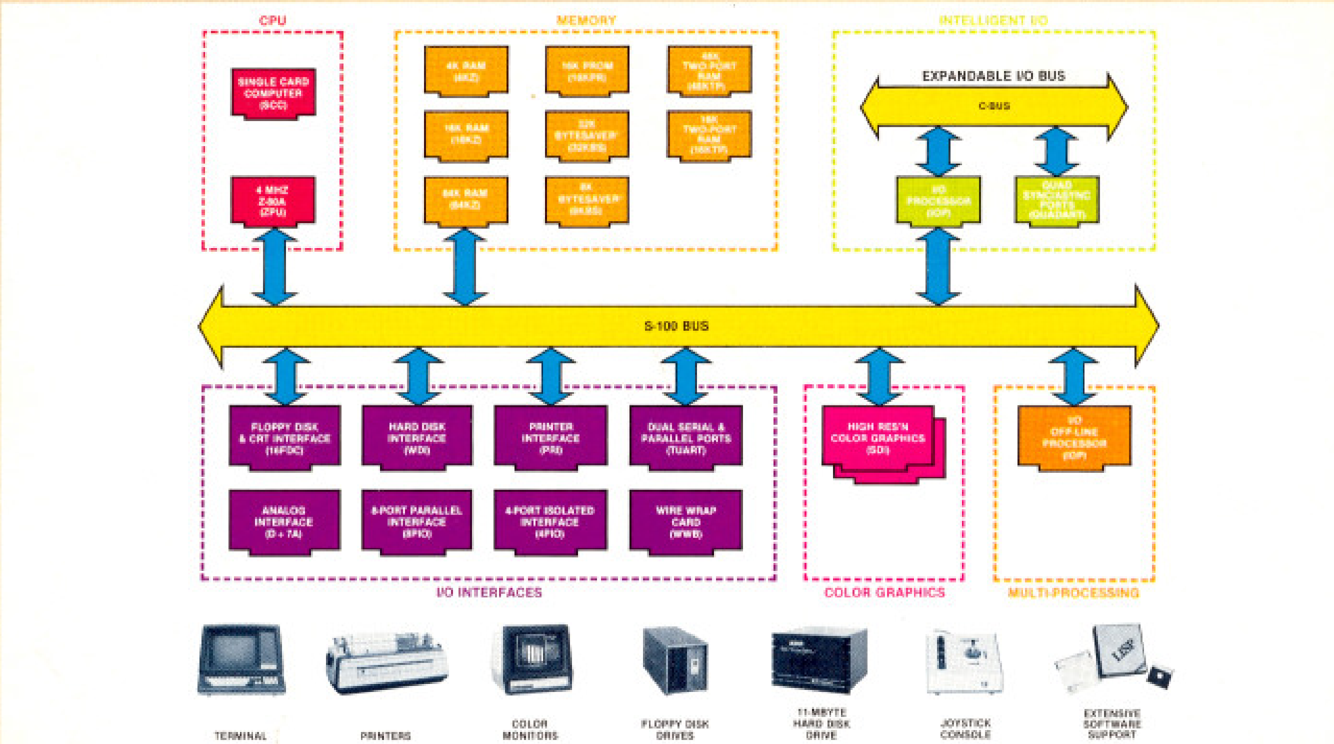
I have written several Assembly language programs that use the disk calls available in CDOS. These work quite well and the operating system takes care of the changes that are due to hardware/software differences between -

1. 5 inch vs. 8 inch disks/drives
2. Single vs. double sided disks/drives
3. Single vs. double density disks/drives

The problem occurs when you do not use the CDOS disk calls for reading the directory. This may occur because you are using an existing CPM program which uses a table for sector interleaving or because you haven't gotten around to writing a routine for disk access. THE PROBLEM IS THAT 5 INCH SINGLE SIDED, SINGLE DENSITY MINI DISKS DO NOT START THEIR DIRECTORY AT SECTOR 1. CDOS knows this and starts properly with track 2 - sector 3 for single sided, single density mini disks which is the fifth interleaved sector. For single sided, single density maxi disks CDOS uses track 2 - sector 1 which is the first interleaved sector.

There are problems with other side/density combinations but I have not had to concern myself with that since I LET CDOS CALLS DO THE WORK. Using CDOS disk calls has an additional benefit in that I wrote an 8 inch single density directory menu routine that automatically calls a language according to the file name extension - IT WORKS ON ANY Cromemco floppy disk system.

Thanks, Jim.
Ed.



What Cromemco computer card capability can do for you

The above diagram shows in a functional way one of the most complete lines of computer cards in the industry.

Look it over carefully. It could be well worth your while.

These are all cards that plug into our S-100 bus microcomputers.

You can also assemble them into a custom system in convenient Cromemco card cages.

MULTI-PROCESSING AND INTELLIGENT I/O

The range of capabilities and versatility you can draw upon is enormous.

In processors, for example, you have a choice of CPU's including our extremely useful new I/O Processor. This can be used as a satellite processor to do off-line processing, multi-processing, and to form intelligent I/O. It opens the door to a whole new group of applications and tasks. Ask us about it.

HIGH RESOLUTION COLOR GRAPHICS

Again, you can have beautiful high-resolution color graphics with our color graphics interface. You can select from over 4000 colors and have a picture with a resolution at least equal to quality broadcast-TV pictures.

You have an unprecedented selection of memory including our unusual 48K and 16K **two-port** RAMs which allow high-speed color graphics.

LOTS OF STORAGE

These days you often want lots of disk storage. So you can select from our disk controller card which will operate our 5" and 8" floppy disk drives (up to 1.2 megabytes). Or select our WDI interface to operate our 11-megabyte hard disk drives.

POWERFUL SOFTWARE AND PERIPHERAL SUPPORT

There's much more yet you can do with our cards. And, of course, there's an easy way to put them to work in our 8-, 12-, and 21-slot card cages. Our PS8 power supply makes it simple to get the system into operation.

Finally, Cromemco offers you the strongest software support in the industry with languages like FORTRAN, C, COBOL, ASSEMBLER, LISP, BASIC and others. There is also a wide choice from independent vendors.

To top it all off, you can draw from a substantial array of peripherals: terminals, printers, color monitors and disk drives.

CONTACT YOUR CROMEMCO REP

There is even more capability than we're able to describe here.

Contact your Cromemco rep now and get this capability working for you.

CROMEMCO COMPUTER CARDS

• **PROCESSORS** — 4 MHz Z-80 A CPU, single card computer, I/O processor • **MEMORY** — up to 64K including special 48K and 16K two-port RAMS and our very well known BYTESAVERS® with PROM programming capability • **HIGH RESOLUTION COLOR GRAPHICS** — our SDI offers up to 754 x 482 pixel resolution. • **GENERAL PURPOSE INTERFACES** —QUADART four-channel serial communications, TU-ART two-channel parallel and two-channel serial, 8PIO 8-port parallel, 4PIO 4-port isolated parallel, D+7A 7-channel D/A and A/D converter, printer interface, floppy disk controller with RS-232 interface and system diagnostics, wire-wrap and extender cards for your development work.

Cromemco ™
i n c o r p o r a t e d
280 BERNARDO AVE., MOUNTAIN VIEW, CA 94040 • (415) 964-7400
Tomorrow's computers today